Distill and Align Decomposition for Enhanced Claim Verification

Jabez Magomere,[1,2] * Elena Kochkina,[2] Samuel Mensah,[2]
Simerjot Kaur,[2] Fernando Acero,[2] Arturo Oncevay,[2]
Charese H. Smiley,[2] Xiaomo Liu,[2] Manuela Veloso[2]
[1]University of Oxford, [2]JPMorgan AI Research
jabez.magomere@keble.ox.ac.uk
{name}.{surname}@jpmorgan.com

Abstract

Complex claim verification requires decomposing sentences into verifiable subclaims, yet existing methods struggle to align decomposition quality with verification performance. We propose a reinforcement learning (RL) approach that jointly optimizes decomposition quality and verifier alignment using Group Relative Policy Optimization (GRPO). Our method integrates: (i) structured sequential reasoning; (ii) supervised finetuning on teacher-distilled exemplars; and (iii) a multi-objective reward balancing format compliance, verifier alignment, and decomposition quality. Across six evaluation settings, our trained 8B decomposer improves downstream verification performance to 71.75% macro-F1, outperforming prompt-based approaches (+1.99, +6.24) and existing RL methods (+5.84). Human evaluation confirms the high quality of the generated subclaims. Our framework enables smaller language models to achieve state-of-the-art claim verification by jointly optimising for verification accuracy and decomposition quality.

1 Introduction

Verifying the factual accuracy of complex long-form text generated by LLMs has become critical as these systems are increasingly deployed in high-stakes applications (Augenstein et al., 2024; Li et al., 2024a; Bang et al., 2025). The *decompose-then-verify* pipeline has emerged as the dominant framework for this task: model responses are split into subclaims, each subclaim is verified against retrieved evidence, and verification results are aggregated to determine overall factuality (Min et al., 2023; Wei et al., 2024; Song et al., 2024).

However, the decomposer can introduce noise such as over-fragmentation, context loss, or insufficient decontextualisation—which harms retrieval and downstream verification (Hu et al., 2025; Liu et al., 2025a). Additionally, decomposers and verifiers are typically optimised independently (Seo et al., 2025), producing subclaims at an atomicity level[1] the verifier may not be trained to handle (Lu et al., 2025). Balancing decomposition quality with verifier alignment is difficult: the verifier's preferred atomicity is latent, and decomposition quality often lacks a reliable, objective metric. Prior work addresses them separately, improving quality with complex multi-stage pipelines that add cost and latency (Liu et al., 2025a; Metropolitansky and Larson, 2025), or improving verifier alignment by deciding only when to decompose without improving the decomposition quality (Lu et al., 2025). We argue that these limitations stem from a fundamental issue: current methods lack a unified training objective that jointly optimizes quality and alignment while remaining computationally efficient.

To address this, we propose a training recipe (shown in Figure 1) with three components. (1) We reframe decomposition as a *sequential reasoning task*, requiring the model to explicitly reason through intermediate steps before generating subclaims. (2) We initialize a decomposer policy via supervised fine-tuning on synthetic decompositions distilled from a larger teacher model. (3) We train the decomposer with GRPO (Shao et al., 2024) using a multi-objective reward that combines verifier feedback for alignment with an explicit decomposition quality checklist.

Across six evaluation settings, our trained 8B decomposer improves downstream verification performance to 71.75% macro-F1, outperforming prompt-based approaches (VeriScore (Song et al., 2024): +1.99pp), existing RL methods (DyDecomp (Lu et al., 2025): +5.84pp), and its own

*Work done during an internship at JPMorgan AI Research.

[1]We follow the atomicity definition from Lu et al. (2025), a metric quantifying information density, atomicity = log_2(# atomic information), where one piece of atomic information is an utterance conveying a single nontrivial fact. Higher atomicity means a claim is more coarse-grained and information-rich.

Findings of the Association for Computational Linguistics: EACL 2026, pages 5887–5912
March 24-29, 2026

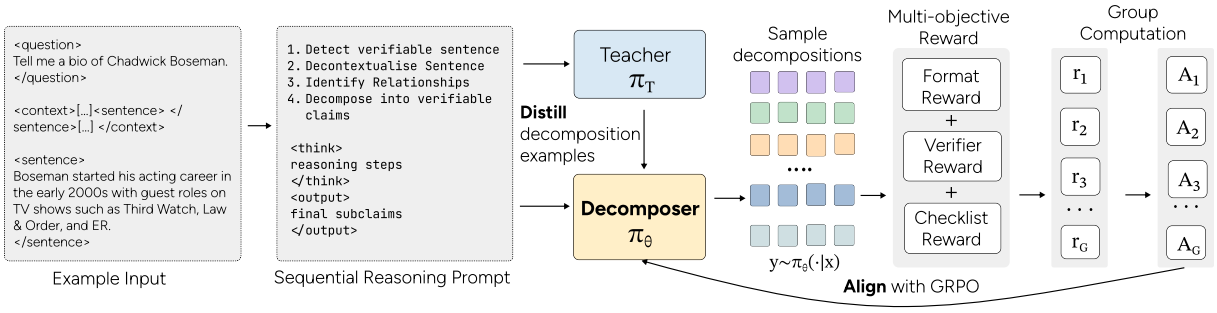

Figure 1: **Distill-Align-Decompose (DAD).** We (1) distill examples from a teacher model, (2) frame decomposition as a sequential reasoning task, and (3) align the policy via GRPO using multi-objective rewards.

prompt-only variant (+2.58pp). Our method generates 8.14 subclaims on average, avoiding both FActScore's (Kamoi et al., 2023) overdecomposition (22.92 subclaims) and DyDecomp's underdecomposition (1.66 subclaims), making the verification process more efficient. Human evaluation confirms high scores across all quality desiderata, validating that our multi-objective training improves both verification accuracy and decomposition quality. We make the following contributions [2]:

- We propose a sequential reasoning framework that structures decomposition in a single model call, improving quality without added latency.
- We introduce a multi-objective GRPO training approach that jointly optimizes verifier alignment and decomposition quality, demonstrating these objectives need not be traded off.
- We conduct human evaluation demonstrating high quality across all desiderata, validating that our approach improves both verification accuracy and decomposition quality.

2 Problem Formulation

The *decompose-then-verify* pipeline is a post-hoc factuality evaluation framework for verifying long-form model responses (Min et al., 2023; Kamoi et al., 2023; Wei et al., 2024; Song et al., 2024; Wanner et al., 2024b). In this framework, each sentence in a response is passed to a decomposer LLM $\mathcal{D}$, whose goal is to generate a set of verifiable subclaims $\{c_1, c_2, \ldots, c_n\}$. Each subclaim c_i is then processed by an evidence retrieval module $\mathcal{R}$, which queries a knowledge source (e.g., Wikipedia or web search API) and returns a set of evidence snippets $E_i = \{e_{i1}, \ldots, e_{ik_i}\}$. The verifier module $\mathcal{V}$ evaluates each claim together with its retrieved evidence and outputs factuality label y_i, where $y_i \in \{\text{SUPPORTED}, \text{NOT SUPPORTED}\}$. The set of claim-level predictions $\{y_i\}$ is then aggregated to produce an overall factuality judgment.

The central challenge in decomposition for accurate factuality verification is to ensure that the generated subclaims are both high quality and aligned with the downstream verifier. Prior work has proposed several desiderata for decomposition quality. Adapting these to the verification setting, we define the task formally: given a sentence S, a decomposer $\mathcal{D}$ generates a set of subclaims $C = \{c_1, c_2, \ldots, c_n\}$, which should satisfy the following properties:

1. **Atomicity Alignment**: Each subclaim c_i is decomposed to the level of granularity expected by the downstream verifier $\mathcal{V}$ (Lu et al., 2025).
2. **Verifiability**: Each subclaim c_i constitutes a verifiable proposition, i.e., "a statement or assertion that can be objectively verified as true or false based on empirical evidence or reality," for which the verifier can assign a label $y_i \in \{\text{SUPPORTED}, \text{NOT SUPPORTED}\}$ (Song et al., 2024).
3. **Entailment**: Each subclaim c_i is entailed by the original sentence ($S \models c_i$) without introducing spurious information (Wanner et al., 2024b).
4. **Coverage**: The set of subclaims $\{c_i\}$ collectively captures all verifiable facts expressed in S (Hu et al., 2025; Metropolitansky and Larson, 2025).
5. **Decontextualization**: Each subclaim c_i is interpretable in isolation, with entities specified and pronouns resolved (Gunjal and Durrett, 2024).

Existing approaches fail to jointly meet these criteria, leaving the challenge of producing subclaims that are both high quality and verifier-aligned. Our proposed approach aims to address this challenge.

[2] Code will be made available upon request.

3 Our Approach

To jointly ensure decomposition quality and verifier alignment, we propose a training recipe with three components presented next.

3.1 Reframing Decomposition as a Sequential Reasoning Task

We reframe decomposition as a step-by-step sequential reasoning task executed within a *single model call per sentence*. Each step conditions the next and requires the decomposer to articulate its reasoning before producing subclaims. This design enforces explicit reasoning about verifiability, decontextualization, relationships, and claim boundaries, while remaining computationally efficient.

Given a model response segmented into sentences $S_1, \ldots, S_n$, we define for each $i \in [n]$ a decomposition input $x_i = (Q, C_i, S_i)$, where Q is the original question (or prompt) corresponding to the model response, and C_i denotes the local context of S_i, consisting of the p preceding and f following sentences, truncated to $[1, n]$. For each target sentence S_i, we construct a local context window including the $p = 2$ preceding and $f = 2$ following sentences. This matches the context window size used by Song et al. (2024), ensuring a fair comparison, but adopts a symmetric split rather than VeriScore's (Song et al., 2024) asymmetric split (3 preceding, 1 following) in order to equally capture both antecedents and consequences of a sentence. The decomposer $\mathcal{D}$ takes as input the triple $x_i = (Q, C_i, S_i)$ and applies a four-step reasoning process below.[3]

1. **Claim detection** ($S_i \rightarrow$ VerifiableSentence): Determine whether S_i contains at least one verifiable proposition (i.e., a fact that can be checked against evidence). If so, return the portion containing verifiable information; otherwise return NO VERIFIABLE CLAIM.
2. **Decontextualization** (VerifiableSentence $\rightarrow$ DecontextualizedSentence): Rewrite the sentence so it is self-contained and unambiguous, resolving references using only Q and C_i. If this is not possible, return CANNOT BE DECONTEXTUALIZED.
3. **Relationship identification** (DecontextualizedSentence $\rightarrow$ [Relationships]): Identify any logical or discourse relations (e.g., attribution, causal, temporal, comparison, expansion, negation, membership) necessary to preserve the meaning of the original sentence.
4. **Claim extraction** (DecontextualizedSentence + Relationships $\rightarrow$ Claims): Split into minimal factual units, each fully decontextualized, verifiable, and interpretable in isolation, while preserving identified relationships.

The decomposer is instructed to return its output y consisting of reasoning steps inside `<think>` tags, and the subclaims inside `<output>` tags.

3.2 Distilling High-Quality Decomposition Exemplars from a Teacher Model

To mitigate cold-start issues before applying RL, we first distill decomposition exemplars from a larger teacher policy π_T and use them to initialize a student policy π_θ. This warm-start serves two purposes: (i) the student acquires instruction-following and output formatting skills prior to RL training, and (ii) the student acquires basic decomposition capabilities from the stronger teacher. The student policy π_θ is trained with a standard supervised fine-tuning (SFT) objective, minimizing the token-level negative log-likelihood of the teacher-generated decomposition examples.

3.3 Aligning the Student Policy via Reinforcement Learning with a Multi-Objective Reward

3.3.1 Reinforcement Learning Formulation

We formulate the problem of aligning the student decomposer π_θ as a single-step Markov Decision Process (MDP). Formally, the state space $x \in \mathcal{S}$ corresponds to the input $x = (Q, C_i, S_i)$, and the action space $y \in \mathcal{A}$ consists of all possible decomposition outputs y (a reasoning trace followed by subclaims). The student policy is parameterized as $\pi_\theta(y \mid x)$, and the reward function $R(x, y)$ is a multi-objective signal combining format, verifier, and checklist reward terms. Each episode is single-step: the agent observes an input x, produces an output $y \sim \pi_\theta(\cdot \mid x)$, and receives the corresponding reward $R(x, y)$. The student policy π_θ is fine-tuned via reinforcement learning to maximize the expected reward $\mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)}[R(x, y)]$ under this formulation.

3.3.2 Group Relative Policy Optimization

We optimize our policy using Group Relative Policy Optimization (GRPO) (Shao et al., 2024). GRPO samples multiple outputs per input and computes group-relative advantages by comparing each

[3] The system prompt used by $\mathcal{D}$ is provided in Appendix K.

sample's reward against the mean reward of its group, which yields more stable gradients when reward distributions vary across input complexities. GRPO also eliminates the need for a separate critic network, making it computationally efficient for fine-tuning large language models.

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \cdot \hat{A}_{\text{group}}(x, y)\Big], \tag{1}$$

where $\hat{A}_{\text{group}}(x, y)$ is the group-relative advantage defined as

$$\hat{A}_{\text{group}}(x, y) = R(x, y) - \frac{1}{|B|} \sum_{(x',y')\in B} R(x', y'), \tag{2}$$

with B denoting a batch of sampled input–output pairs and $R(x, y)$ the reward function.

3.3.3 Multi-Objective Reward Function

To align the policy π_θ with our decomposition desiderata and the downstream verifier, we design a multi-objective reward function comprising three reward terms: a format reward for structured output, a verifier reward to capture atomicity preferences, and a checklist reward for subclaim quality. During training, GRPO samples completions $y_1, \ldots, y_k \sim \pi_\theta(\cdot|x)$ for each input x and scores them with these reward terms to compute group-relative advantages. Each reward term is normalized to $[0, 1]$, defined as follows:

Format Reward. Correct formatting is essential for downstream parsing: reasoning steps must appear in `<think></think>` tags and subclaims in `<output>[]</output>` tags as a valid list. We use a soft reward with partial credit (rather than binary pass/fail) to provide more informative gradients for the policy. The format reward is computed as $r_{\text{format}}(y) = \sum_i w_i \cdot f_i(y)$, where $f_i(y) \in \{0, 1\}$ are binary checks for: presence of both required tags, correct tag ordering, successful list parsing, and non-empty list with no blank strings, and w_i is a weight assigned to each check (Appendix I). We use a soft format reward instead of structured generation to maintain compatibility with RL training, reduce training overhead, and draw on prior evidence that soft rewards provide richer learning signals (Damani et al., 2025; Shao et al., 2024).

Verifier Reward. Since the verifier's preferred atomicity level cannot be directly modeled, we use verifier prediction accuracy as a proxy: outputs that yield accurate verifier predictions reveal the appropriate subclaim atomicity for a given verifier. For each output, we extract subclaims and retrieve k supporting evidence snippets from a knowledge source for each subclaim c_i. A verifier model $\mathcal{V}$ scores each claim–evidence pair to obtain factuality probabilities $p_i \in [0, 1]$, which are aggregated into a sentence-level prediction and compared to the ground-truth label $\ell \in \{0, 1\}$ of sentence S_i from the input. We compare two verifier reward designs:

1. **Sparse.** Aggregate claims via a logical AND: predict SUPPORTED if and only if all $p_i \geq \tau$. Let $\hat{\ell} \in \{0, 1\}$ denote the resulting sentence-level prediction. The sparse verifier reward is $r_{\text{ver}}(x, y) \in \{0, 1\}$, where $r_{\text{ver}}(x, y) = 1$ if $\hat{\ell} = \ell$ and 0 otherwise.
2. **Dense.** Following Damani et al. (2025), we replace binary accuracy with the Brier score: for predicted probability $q \in [0, 1]$ and outcome $o \in \{0, 1\}$, $\text{Brier}(q, o) = (q - o)^2$. For a list of subclaims with verifier probabilities $p_1, \ldots, p_m$, we compute the sentence-level probability $\bar{p}$ via the geometric mean, which is more sensitive to low confidences. The dense verifier reward is $r_{\text{ver}}(x, y) = 1 - (\bar{p} - \ell)^2$.

Checklist Reward. To ensure high-quality decompositions, we introduce a weighted checklist rubric that evaluates each decomposed subclaim against multiple criteria. We employ an LLM-as-a-Judge $\mathcal{J}$, which assesses whether each criterion c_i is satisfied for each subclaim. The judge outputs a binary score for each criterion, $\mathcal{J}(x, y, c_i) \to \{0, 1\}$, and the checklist reward for each subclaim is calculated as the weighted average $\frac{\sum_i w_i \cdot \mathcal{J}(x,y,c_i)}{\sum_i w_i}$. The final checklist reward $r_{\text{checklist}}(x, y)$ is the geometric mean across all subclaims. The criteria cover: complete verifiability (single verifiable proposition), retrieval relevance with respect to the question, presence of all necessary qualifiers, explicit references to entities and relations, and absence of ungrounded additions. The complete checklist with detailed criteria and prompts is in Appendix I.

Overall Reward. For an input $x \sim \mathcal{D}$ and completion $y \sim \pi_\theta(\cdot \mid x)$, the reward function is the equally-weighted sum of our three reward terms:

$$R(x, y) = r_{\text{format}}(y) + r_{\text{ver}}(x, y) + r_{\text{checklist}}(x, y).$$

4 Experimental Setup

4.1 Datasets

SFT. We build the SFT dataset from open-model responses on long-form factuality benchmarks (VeriScore (Song et al., 2024) and VeriFastScore (Rajendhran et al., 2025a)). We sample a balanced dataset of **15.7k** sentences with surrounding context from these responses, varying prompt source, model source, and sentence lengths. We then use the teacher model with our decomposition prompt to generate synthetic decompositions. After filtering malformed outputs, we obtain **13.7k** training examples and **1.5k** test examples.

RL. For RL training, we require a dataset containing sentence-level factuality labels to compute the verifier reward. We use the dataset constructed by Lu et al. (2025), that contains factuality-annotated sentences across varying input atomicity for responses from ChatGPT and PerplexityAI to FActScore (Min et al., 2023). To align with Lu et al. (2025) and expose the policy to more complex claims, we retain *Atomicity 1* (single-sentence units) and *Atomicity 2* (multi-sentence spans), producing **2.3k** training instances. We use the Wikipedia dump from Min et al. (2023) as the knowledge source.

Evaluation Datasets. We evaluate across four datasets under six evaluation settings that vary in input granularity, domain, and knowledge source (Table 4). We group settings by annotation granularity: (1) *sentence-level* datasets with SUPPORTED/NOT SUPPORTED labels for individual claim units, and (2) *response-level* datasets with a factuality label for the entire response. For sentence-level evaluation, we use test split from Lu et al. (2025), consisting of **ChatGPT** and **PerplexityAI** responses to FActScore biography prompts (Min et al., 2023). We evaluate two input granularities for each source: *Atomicity-1* (single sentences) and *Atomicity-2* (multi-sentence spans). Evidence retrieval uses top-5 passages from Wikipedia. For response-level evaluation, we use two datasets with Google Search as knowledge source. **FELM** (Chen et al., 2023) contains annotated ChatGPT responses across multiple domains, focusing on world knowledge questions. **BINGCHAT** (Li et al., 2024b) contains factuality annotations for Microsoft Copilot responses spanning diverse topics; these responses are significantly longer than other datasets. For both datasets, we retrieve top-10 results from Google Search via SerpAPI[4] as evidence.

4.2 Models

We conduct all experiments with the Llama family of models (Grattafiori et al., 2024). For the **decomposer model**, we use **Llama-3.1-8B-Instruct**. For the **verifier**, we use **Bespoke-MiniCheck-7B** (Tang et al., 2024)[5], a specialised fact-verification model that achieves state-of-the-art performance on aggregated fact-checking benchmarks while being computationally efficient, making it suitable for reward modelling tasks. As the **teacher model**, we use **Llama-3.1-405B-Instruct** to generate the synthetic data for the SFT warm-up. For the **LLM-as-a-Judge**, we use **Llama-3.3-70B-Instruct**, which provides reliable evaluations at a favourable quality–cost trade-off.

Baselines. We compare our trained decomposer to prompt-based, SFT-based, and RL-based baselines, and probe model-scale effects on verification accuracy. For **prompt baselines** using Llama-3.1-8B-Instruct, we compare against two few-shot approaches; **FActScore** (Min et al., 2023) (extracts atomic facts) and **VeriScore** (Song et al., 2024) (focuses on verifiability), and our decomposition prompt applied zero-shot to ablate training gains. For the **SFT baseline**, we compare against the open-source VeriScore claim extractor (Song et al., 2024) (Mistral-7B), trained on 13.4k GPT-4–distilled decompositions. For the **RL baseline**, we implement DyDecomp (Lu et al., 2025), a PPO-based approach that trains a policy to dynamically determine when to decompose claims using verifier confidence as a reward. We preserve their configuration (Llama-3-70B-Instruct decomposer, Llama-3-8B-Instruct verifier) for faithful comparison. Unlike all other baselines which segment responses into sentences before decomposition, DyDecomp operates directly on full responses. To probe model scale effects, we run our zero-shot prompt on Llama-3.3-70B-Instruct and Llama-3.1-405B-Instruct. Implementation details (training and hyper-parameters) of our experiments are provided in Appendix H.

Metrics. For sentence-level datasets, we follow Lu et al. (2025): a claim is SUPPORTED if and only if all of its subclaims are SUPPORTED (logical AND). For response-level datasets, following

[4] https://www.serpapi.com

[5] https://huggingface.co/bespokelabs/Bespoke-MiniCheck-7B

Decomposer	Approach	PerplexityAI						ChatGPT						FELM			BINGCHAT			Overall		
		Atomicity-1			Atomicity-2			Atomicity-1			Atomicity-2											
		BAcc	F1	SC	BAcc	F1	SC	BAcc	F1	SC	BAcc	F1	SC	BAcc	F1	SC	BAcc	F1	SC	BAcc	F1	SC
Baselines																						
Llama-3.1-8B	FActScore	76.10	65.72	10.76	71.40	67.07	16.66	72.60	73.68	11.08	61.60	64.33	17.55	53.93*	64.04*	8.9*	50.24*	58.24*	72.56*	64.31	65.51	22.92
Mistral-7B	VeriScore-SFT	73.50	65.61	2.50	66.60	63.63	4.46	77.40	77.69	2.68	76.90	75.79	4.90	52.59	66.40	4.39	51.51	64.23	21.26	66.42	68.89	6.70
Llama-3.1-8B	VeriScore	78.60	71.61	2.49	71.00	68.48	4.20	82.30	82.76	2.85	73.70	75.84	5.76	**59.56***	67.86*	6.65*	52.15*	52.02*	28.00*	69.55	69.76	8.33
Llama-3-70B	DyDecomp	67.52	67.33	1.62	60.53	60.23	1.66	76.38	72.57	1.59	77.42	59.17	1.64	55.28	68.55	1.73	50.86	**67.60**	1.72	64.67	65.91	1.66
DAD Prompt Variants																						
Llama-3.1-8B	Prompt	73.20	67.46	2.00	69.70	68.23	3.15	79.73	79.73	2.09	75.90	71.04	3.50	56.28	**68.57**	5.05	**56.28**	60.00	30.73	68.52	69.17	7.75
Llama-3.3-70B	Prompt	76.60	70.45	2.47	72.90	**71.13**	4.22	**84.70**	**85.01**	2.60	77.90	77.93	4.73	53.38	65.81	5.06	53.38	57.48	30.10	69.81	71.30	8.20
Llama-3.1-405B	Prompt	76.80	69.61	2.33	71.30	68.51	4.20	82.00	82.13	2.27	82.70	77.27	4.71	55.10	67.23	5.16	55.10	61.67	33.59	70.50	71.07	8.71
Ours																						
Llama-3.1-8B	DAD (Ours)	**79.70**	**73.15**	2.36	**73.30**	70.59	3.87	81.70	81.74	2.41	**81.60**	**78.55**	4.26	56.37	67.23	5.22	55.53	59.21	30.73	**71.37**	**71.75**	8.14

Table 1: Balanced Accuracy (BAcc), macro-F1, and subclaim count (SC) per dataset. Best BAcc and F1 in each column are **bold**. The final block reports overall BAcc/F1 and the average SC. Entries marked *are reported from Hu et al. (2025) under a similar setup. **On average across six evaluation settings, our 8B decomposer leads to the highest verification performance, with BAcc** 71.37% **and macro-F1** 71.75%**, while maintaining comparable granularity (avg.** 8.14 **subclaims).**

Hu et al. (2025), we aggregate subclaim-level verifier scores via harmonic mean; a response is SUPPORTED when the aggregate exceeds 0.5. We report balanced accuracy (BAcc) amd macro-F1 over {SUPPORTED, NOT SUPPORTED} to handle class imbalance. We also report the average subclaim count per input to quantify downstream computational cost: each subclaim requires independent evidence retrieval and verifier inference, so higher subclaim counts translate directly to increased latency and compute requirements.

5 Results & Discussion

We present our results in Table 1, evaluating decomposition approaches across six settings varying input granularity, knowledge sources, and domains.

Overall performance. Our Llama-3.1-8B DAD decomposer achieves the highest overall verification performance, with 71.37% balanced accuracy and 71.75% macro-F1 averaged across six evaluation settings. DAD achieves the best performance on three settings (PerplexityAI Atomicity-1, PerplexityAI Atomicity-2 BAcc, and ChatGPT Atomicity-2); larger prompted models perform best on ChatGPT Atomicity-1, and VeriScore or prompt baselines lead on FELM and BINGCHAT. These benchmarks differ substantially in domain, response length, and evidence sources, and prior work shows that decomposition methods often behave inconsistently across datasets due to varying evidence granularity requirements (Hu et al., 2025). Despite this variability, DAD is the only decomposer that achieves the best aggregate performance across all settings. The largest gains over prompt-only baselines occur on ChatGPT Atomicity-2 (↑+7.51 F1) and PerplexityAI Atomicity-1 (↑+5.69 F1), showing that specialized training yields meaningful improvements for decomposition-based verification. While absolute gains vary by dataset, reflecting differences in claim atomicity, evidence characteristics, and retrieval difficulty, DAD improves overall performance relative to all baselines.

Comparison with prompt-based methods. Our trained decomposer improves verification performance compared to prompt-based approaches. Relative to FActScore, which produces an average of 22.92 subclaims, our method achieves ↑+6.24pp F1 (71.75% vs. 65.51%) while generating substantially fewer subclaims (8.14 vs. 22.92). This pattern suggests that over-decomposition may introduce noise, consistent with observations in prior work (Hu et al., 2025). Compared to VeriScore (69.76% F1, 8.33 subclaims), our approach achieves ↑+1.99pp F1 with comparable granularity (8.14 subclaims), indicating that our training procedure produces more effective decompositions at comparable granularity levels. The improvement increases to ↑+7.19pp F1 on BINGCHAT, where both methods face challenges from longer responses (generating ∼30 subclaims). Appendix Table 10 provides examples of decomposition by different methods, highlighting some of the issues.

Comparison with RL-Based Methods. Our approach outperforms DyDecomp (Lu et al., 2025) overall (↑+7.09 BAcc, ↑+5.84 macro-F1). On BINGCHAT, however, DyDecomp achieves higher F1 (67.60%) but exhibits severe class imbalance: 93.69% recall on SUPPORTED versus only 8.02% on NOT SUPPORTED (identifying fewer

than 1 in 10 unsupported claims). Our decomposer maintains balanced recall (61.06% / 50.00%; ↑+41.98 on NOT SUPPORTED vs. DyDecomp) despite lower F1 (59.21%). DyDecomp trains a policy to decide whether to decompose at the *response level* (no prior sentence segmentation), using verifier confidence as reward. This leads to minimal decompositions (1.66 subclaims on average), which appears to favour the SUPPORTED class: responses containing predominantly supported claims are easier to verify even with underdecomposed subclaims. However, NOT SUPPORTED responses typically contain mixtures of supported and unsupported claims (Wei et al., 2024). Without fine-grained decomposition that isolates specific propositions, retrieval is less likely to surface disconfirming evidence for NOT SUPPORTED cases. Our fixed sentence segmentation maintains consistent granularity, enabling balanced discrimination across both classes, which is critical for optimal verification.

Effect of model scale. Our fine-tuned 8B model achieves 71.75% overall F1, comparable to prompted larger models: Llama-3.3-70B (71.30%) and Llama-3.1-405B (71.07%). Notably, scaling from 70B to 405B parameters yields only +0.23pp F1 improvement when using the same prompt, while task-specific training of the 8B model provides ↑+2.58pp improvement over its prompt-only variant (71.75% vs. 69.17%). This pattern suggests diminishing returns to scale for prompt-only decomposition, or low sensitivity of the verifier pipeline to finer subclaim granularity. Either way, specialised training enables a smaller model to reach competitive aggregate performance.

Experiment with a non-aligned verifier. Our main results optimize the DAD decomposer for a specific verifier (Bespoke-MiniCheck-7B). To test whether DAD's decompositions generalize beyond the aligned verifier, we pair it with ClearCheck-8B (Seo et al., 2025) and compare against a baseline using Llama-8B with the same prompt. As shown in Table 2, DAD-ClearCheck outperforms Llama-8B-ClearCheck on most datasets, confirming that DAD provides value even without verifier-specific alignment. Performance remains below the aligned DAD-MiniCheck combination, indicating that verifier-specific optimization still yields the best results. The MacroF1 drop on BingChat arises from long claims (averaging 30+ subclaims) and ClearCheck's high-confidence outputs, which tend to over-predict NOT SUPPORTED. Performance may also be limited by using a study-wide prompt rather than a verifier-specific prompt.

Dataset	Metric	DAD ClearCheck	Llama-8B ClearCheck
PerplexityAI Atom.-1	BAcc	**72.50**	69.50
	MacroF1	**67.00**	66.50
PerplexityAI Atom.-2	BAcc	**71.70**	59.70
	MacroF1	**70.80**	59.80
ChatGPT Atom.-1	BAcc	**79.60**	66.60
	MacroF1	**79.90**	65.40
ChatGPT Atom.-2	BAcc	65.90	**69.70**
	MacroF1	**63.30**	57.50
FELM	BAcc	**56.95**	53.84
	MacroF1	**62.50**	61.75
BINGCHAT	BAcc	51.54	**53.17**
	MacroF1	18.70	**24.90**

Table 2: DAD vs. Llama-8B decomposers paired with a non-aligned ClearCheck-8B verifier. **DAD consistently outperforms Llama-8B, demonstrating that the decomposer provides value even without verifier-specific alignment**. Bold indicates the higher score per dataset and metric.

Training Configuration	PerplexityAI		FELM	
	F1	SC	F1	SC
Prompt only (no training)	67.46	2.00	**68.57**	5.05
RL only (no SFT)				
+ Format + Verifier Accuracy (sparse)	70.86	1.95	58.94	3.78
+ Format + Verifier Brier (dense)	72.71	1.49	64.94	3.40
SFT only	71.55	2.28	66.39	5.01
SFT + RL (partial)				
+ Format + Verifier Brier	71.70	2.03	66.10	4.43
+ Format + Checklist	69.90	2.28	67.50	4.97
Full SFT + RL framework (ours)				
+ Format + Verifier Brier + Checklist	**73.15**	2.36	67.23	5.22

Table 3: Ablation study on different training setups. We report macro-F1 and Average Subclaim Count (Avg. SC) on PerplexityAI and FELM , plus overall F1. Best F1 per column is **bold**.

6 Ablation Study

To evaluate each component's impact on verification macro-F1, we perform an ablation study on PerplexityAI (*Atomicity-1*) and FELM datasets (Table 3). Our complete training approach—combining SFT warm-up and multi-objective rewards—yields the best overall verification performance. We highlight two key insights:

Dense Brier rewards yield more sample-efficient and stable learning. Figure 2 shows that the base decomposer policy trained with the dense Brier reward achieves higher reward levels with fewer training steps compared to the sparse accuracy reward, while also exhibiting more stable learning dynamics and higher reward at plateau. The

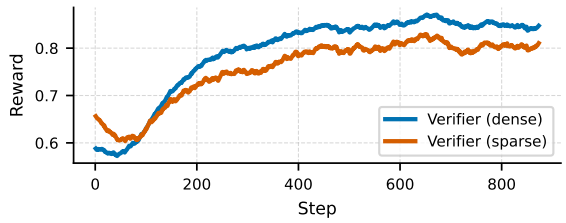

Figure 2: **Verifier training rewards (dense vs. sparse).** Training the base policy with the verifier Brier reward (● dense) yields smoother, more sample-efficient learning than the sparse accuracy reward (● sparse).

Brier objective leverages verifier confidence, penalizing the policy when decompositions lead to low-confidence predictions, whereas the sparse reward treats all instances equally. This richer shaping signal improves sample efficiency and translates into stronger downstream verification performance, with the dense-reward policy outperforming its sparse-reward counterpart on both PerplexityAI (72.71% vs. 70.86%) and FELM (64.94% vs. 58.94%; Table 3).

Verifier-only reward optimisation biases the policy toward coarser decompositions, reducing generalisation. Table 3 shows that policies optimised solely with verifier rewards produce substantially fewer subclaims, indicating a tendency toward coarse decompositions. While this aligns the policy more closely with verifier preferences on the training distribution, it reduces performance on out-of-domain data: both verifier-only setups underperform on FELM relative to the base model (e.g., 64.94% vs. 68.57% for dense; 58.94% vs. 68.57% for sparse). These results suggest that relying exclusively on verifier feedback biases the policy toward in-domain verifier behaviour at the expense of generalisation. Although verifier feedback is valuable, optimising on it alone can be detrimental, and the best cross-dataset performance arises when it is combined with SFT and checklist objectives to regularise decomposition granularity. We provide further verification error analysis and examples in Appendix F.

Additionally, we qualitatively analyzed our reasoning prompt's contribution, with example traces provided in Appendix G. Each decomposition step meaningfully contributes to accurate claim extraction, with intermediate reasoning (e.g., filtering non-verifiable statements, resolving pronouns, identifying relationships) systematically improving final outputs.

7 Decomposition Quality Estimation

While recent *decompose-then-verify* frameworks rely on generated subclaims from LLMs, the quality of these decompositions is not guaranteed. To assess the quality of decompositions across the desiderata for decomposition quality (see Section 2), we manually annotated a subset of claims from our datasets, evaluating decompositions generated by our methodology and compared approaches (annotation guidelines in Appendix A).

Specifically, we sampled 25 model responses across datasets, which are decomposed using our method DAD, FActScore, LLama-405B and Veriscore. These yield 429 sentences in total for annotation at the subclaim-level. We conduct expert annotations across the 5 desiderata: verifiability, coherence, clarity, completeness and uniqueness. Three annotators (co-authors) with expertise in natural language processing independently labeled each subclaim (or subclaim set) with binary judgement (1/0). To prevent bias, annotators were blinded to the decomposition method. Annotation agreement across the five desiderata was generally strong, with three-way agreement rates ranging from 74% to 93% and Fleiss' Kappa values between 0.5 and 0.88.[6] We apply majority vote to determine the final labels. For any given model and desideratum (except completeness and uniqueness), scores are averaged across subclaims of the sentence. Completeness and uniqueness are assigned a single binary score at the sentence-level. See annotated examples in Appendix Table 6.

Discussion Figure 3 presents the results. Our method which uses Llama-8B achieves a high average score for the different desiderata, demonstrating that our generated subclaim decompositions are well-formed. Notably, being significantly smaller, our method outperforms Llama-405B-Prompt across most dimensions. Interestingly, FactScore achieves high completeness, matching our method and VeriScore, but exhibits near-zero uniqueness. This suggests it generates overlapping subclaims to ensure coverage. While this might maximize information coverage, it introduces numerous trivially obvious subclaims (e.g., "12 Feb. 2025 is a date") or may contain information that is not directly relevant to the original claim. Incoherent subclaims provide little verification value, while redundant subclaims cause increased costs

[6] Detailed inter-annotator agreement results are shown in Appendix Table 5.

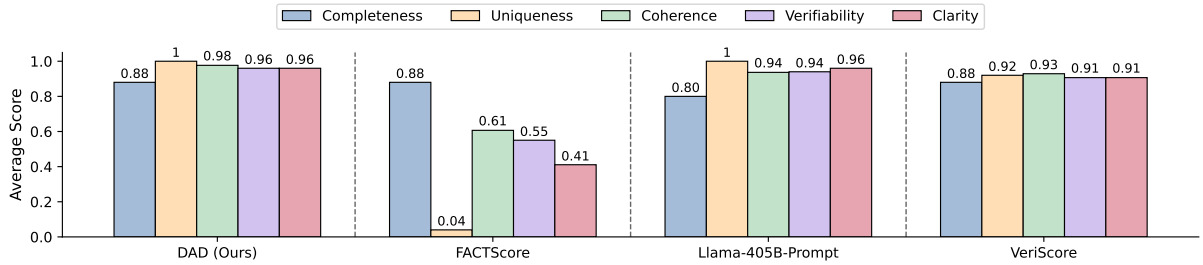

Figure 3: **Human-evaluated subclaim decomposition quality across the five desiderata.** Our method achieves comparable completeness to baselines while maintaining high scores across other desiderata. Scores represent sentence-level averages of subclaim quality with higher values indicating better quality for a given desideratum.

via retrieval and verifier calls.

8 Related Works

Fact verification of long-form *LLM responses* commonly follows a *decompose–then–verify* pipeline, in which model outputs are split into subclaims that are checked against evidence (Min et al., 2023; Wei et al., 2024). Existing decomposition approaches target different objectives; atomicity (Min et al., 2023; Wei et al., 2024), verifiability (Song et al., 2024), decontextualisation (Gunjal and Durrett, 2024), and efficiency (Rajendhran et al., 2025a)—but are primarily designed for benchmarking LLM factuality rather than accurate verification. This yields two key limitations: **verifier misalignment** (decomposition objectives are not optimised with respect to the verifier, which is treated as fixed; Lu et al., 2025) and **decomposition quality** (decomposition can introduce noise that degrades downstream verifier accuracy; Hu et al., 2025).

Recent work addresses these limitations separately. To improve decomposition quality, Wanner et al. (2024a); Gunjal and Durrett (2024) focus on generating *molecular facts*—i.e., minimal, fully decontextualised subclaims; Liu et al. (2025a) aim to capture missing or incomplete facts; and Metropolitansky and Larson (2025) propose a multi-stage pipeline that separates selection, disambiguation, and extraction. While these methods improve different decomposition *desiderata*, they do not address verifier alignment and often require complex pipelines with closed-source models. Conversely, Lu et al. (2025) tackle verifier misalignment by training, via PPO, a separate policy that decides whether to decompose based on verifier confidence as reward. However, this approach learns only *when* to decompose while leaving the decomposer unchanged, achieving alignment without improving decomposition quality. **Our approach addresses both limitations jointly**. We reframe decomposition as a sequential reasoning task in which the decomposer performs explicit intermediate reasoning steps before generating subclaims within a single model call. To achieve verifier alignment *and* high-quality decomposition, we propose a **multi-objective reward** that combines a verifier-based signal with an explicit decomposition-quality checklist, and we train the decomposer policy with GRPO (Shao et al., 2024). This joint optimisation yields consistent gains in verification accuracy and in human-evaluated decomposition quality. Appendix E provides a comparative table of our method versus prior work, emphasizing novelty.

9 Conclusion

We address the challenge of complex claim verification by jointly optimizing decomposition for quality and verifier alignment. Our novel multi-objective reward, combined with structured sequential reasoning, enables efficient generation of high-quality subclaims and improves verification accuracy. Our trained decomposer improves verification accuracy while maintaining balanced subclaim granularity across diverse evaluation settings.

Limitations

While our approach demonstrates improvements in claim decomposition and verification, several limitations remain. First, we only test our approach on a fixed verifier and are limited by the available reference datasets, which impacts the extent to which we can improve and generalize verification accuracy. Future work should assess the framework across a broader range of verifiers and knowledge sources. Second, supervised pre-training relied on synthetic exemplars from a single teacher model, potentially introducing bias and limiting decomposition diversity. Third, our multi-objective reward

and LLM-as-a-Judge evaluations, while scalable, may not fully capture nuanced aspects of decomposition quality; human evaluation was limited in scope. Fourth, our LLM-as-a-Judge element may introduce some undesired bias in the training process, which could be assessed by further experimentation with alternative large models as judges. Fifth, the approach was tested primarily on English-language data and Wikipedia/Google Search, leaving multilingual and domain-specific verification as future work. Sixth, our evaluation did not include multi-hop claim verification benchmarks, which are important for assessing performance on complex reasoning tasks involving multiple interconnected facts. Lastly, sentence-level segmentation may not optimally capture all factual relationships, especially those spanning multiple sentences or extend across the entire document. Investigating more adaptive segmentation and decomposition strategies could further improve performance on complex long-form texts.

Ethical Considerations

Our work uses publicly available datasets and does not involve private or sensitive data. However, factuality and neutrality of sources like Wikipedia and web search are not guaranteed, and any biases or inaccuracies may affect system outputs. Automated fact verification is not infallible; errors in decomposition or verification could lead to incorrect judgments, especially in sensitive domains. Our models may also inherit social biases from training data, and we do not explicitly address bias mitigation. Human evaluation was conducted by expert annotators following clear guidelines; annotators were blinded to system identity to reduce bias, and no personally identifiable information was involved. Finally, our experiments focus on English-language data, and ethical implications may differ in other languages or cultural contexts.

Acknowledgments

We thank Uljad Berdica, Caiqi Zhang, Salim Amoukou and Myeong-jun Erik Jang for insightful discussions and valuable feedback.

References

Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, and 1 others. 2024. Factuality challenges in the era of large language models and opportunities for fact-checking. *Nature Machine Intelligence*, 6(8):852–863.

Yejin Bang, Ziwei Ji, Alan Schelten, Anthony Hartshorn, Tara Fowler, Cheng Zhang, Nicola Cancedda, and Pascale Fung. 2025. HalluLens: LLM hallucination benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24128–24156, Vienna, Austria. Association for Computational Linguistics.

William Brown. 2025. Verifiers: Environments for llm reinforcement learning. `https://github.com/willccbb/verifiers`.

Shiqi Chen, Yiran Zhao, Jinghan Zhang, I-Chun Chern, Siyang Gao, Pengfei Liu, and Junxian He. 2023. Felm: Benchmarking factuality evaluation of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. 2025. Beyond binary rewards: Training lms to reason about their uncertainty. *Preprint*, arXiv:2507.16806.

Michael Han Daniel Han and Unsloth team. 2023. Unsloth.

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.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Qisheng Hu, Quanyu Long, and Wenya Wang. 2025. Decomposition dilemmas: Does claim decomposition boost or burden fact-checking performance? 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 6313–6336, Albuquerque, New Mexico. Association for Computational Linguistics.

Ryo Kamoi, Tanya Goyal, Juan Diego Rodriguez, and Greg Durrett. 2023. WiCE: Real-World Entailment for Claims in Wikipedia. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7561–7583, Singapore. Association for Computational Linguistics.

Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024a. The dawn after the dark: An empirical study on factuality hallucination in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10879–10899, Bangkok, Thailand. Association for Computational Linguistics.

Miaoran Li, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhu Zhang. 2024b. Self-checker: Plug-and-play modules for fact-checking with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 163–181, Mexico City, Mexico. Association for Computational Linguistics.

Xin Liu, Lechen Zhang, Sheza Munir, Yiyang Gu, and Lu Wang. 2025a. Verifact: Enhancing long-form factuality evaluation with refined fact extraction and reference facts. *Preprint*, arXiv:2505.09701.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*.

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.

Dasha Metropolitansky and Jonathan Larson. 2025. Towards effective extraction and evaluation of factual claims. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6996–7045, Vienna, Austria. 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.

Rishanth Rajendhran, Amir Zadeh, Matthew Sarte, Chuan Li, and Mohit Iyyer. 2025a. Verifastscore: Speeding up long-form factuality evaluation. *Preprint*, arXiv:2505.16973.

Rishanth Rajendhran, Amir Zadeh, Matthew Sarte, Chuan Li, and Mohit Iyyer. 2025b. VeriFastScore: Speeding up long-form factuality evaluation. *arXiv preprint*. ArXiv:2505.16973 [cs].

Wooseok Seo, Seungju Han, Jaehun Jung, Benjamin Newman, Seungwon Lim, Seungbeen Lee, Ximing Lu, Yejin Choi, and Youngjae Yu. 2025. Verifying the verifiers: Unveiling pitfalls and potentials in fact verifiers. In *Second Conference on Language Modeling*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Yixiao Song, Yekyung Kim, and Mohit Iyyer. 2024. VERISCORE: Evaluating the factuality of verifiable claims in long-form text generation. *arXiv preprint*. ArXiv:2406.19276 [cs].

Liyan Tang, Philippe Laban, and Greg Durrett. 2024. MiniCheck: Efficient fact-checking of LLMs on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8818–8847, Miami, Florida, USA. Association for Computational Linguistics.

Miriam Wanner, Benjamin Van Durme, and Mark Dredze. 2024a. Dndscore: Decontextualization and decomposition for factuality verification in long-form text generation. *Preprint*, arXiv:2412.13175.

Miriam Wanner, Seth Ebner, Zhengping Jiang, Mark Dredze, and Benjamin Van Durme. 2024b. 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.

A Annotation Guidelines

A.1 Overview

This framework evaluates decomposed subclaims on five qualitative dimensions. Each dimension is scored in a binary fashion (1/0) per annotator. Scores are averaged across dimensions to yield a subclaim-level score, and across subclaims to yield model-level performance.

A.2 Dimensions

Verifiability (Specificity) Can the subclaim be checked against a reliable reference (e.g., Google Search)?

- **Score 1**: Verifiable.
- **Score 0**: Vague or unverifiable.

Examples:

- "Tesla's Q4 2024 revenue was $25 billion." (**Verifiable**)
- "Tesla is very successful." (**Not verifiable**)

Faithfulness (Coherence / Entailment) Does the subclaim preserve the meaning of the original sentence without adding, removing, or distorting information? If the original sentence encodes a relationship (causal, temporal, comparative, conditional, attribution), the decomposition must preserve that relationship.

- **Score 1**: Fully faithful to the source.
- **Score 0**: Alters or misrepresents the source or does not preserve relationship.

Examples:

- **Original claim**: The Fed raised interest rates by 0.25% in March.
  - **Subclaim**: The Fed raised interest rates in March. (**Faithful simplification**)
  - **Subclaim**: The Fed raised interest rates by 0.5% in March. (**Distorted detail**)
- **Original claim**: James was born in 1998.
  - **Bad Decomposition**: James was born. 1998 exists.
  - **Good Decomposition**: James was born in 1998. (Temporal relationship preserved)

Clarity (Self-sufficiency / Decontextualization + Non-triviality) Is the subclaim clear, unambiguous, and understandable without additional context?

- **Score 1**: Self-contained and precise.
- **Score 0**: Ambiguous (e.g., pronouns, vague references) or trivial (e.g., "X exists").

Examples:

- "Microsoft acquired Activision Blizzard in 2022." (**Clear**)
- "They acquired the company in 2022." (**Unclear reference**)
- "James is born." (**Trivial**)
- "1998 exists." (**Trivial**)

Coverage (Completeness) Does the set of subclaims for a sentence capture all verifiable facts expressed in the original sentence?

- **Score 1**: All facts are captured.
- **Score 0**: One or more facts are missing.

Examples:

- **Original**: California and New York implemented plastic bag bans.
- **Subclaims** (Coverage = 1):
  - California implemented a plastic bag ban.
  - New York implemented a plastic bag ban.
- **Subclaims** (Coverage = 0; New York fact missing):
  - California implemented a plastic bag ban.

Uniqueness Do the extracted subclaims avoid duplicating the same fact in different word orders or paraphrases?

- **Score 1**: Each subclaim adds unique factual content, with no overlap in meaning.
- **Score 0**: One or more subclaims repeat the same fact (e.g., "A and B" vs. "B and A").

Examples:

- **Original**: California and New York implemented a plastic bag ban.

- **Subclaims** (Unique facts; Uniqueness score = 1):
  - California implemented a plastic bag ban.
  - New York implemented a plastic bag ban.
- **Original**: Microsoft and Google released AI chatbots.
- **Subclaims** (Duplication; Uniqueness score = 0):
  - Microsoft released an AI chatbot.
  - Google released an AI chatbot.
  - AI chatbots were released by Microsoft and Google.

A.3 Scoring Methodology

Each subclaim receives a score for Verifiability, Faithfulness, and Clarity. All subclaims together for each original sentence receive a Coverage and Uniqueness score.

B Dataset Details

Granularity	Dataset	Atomicity	Size	KS
Sentence	PerplexityAI	1	300	Wiki
	ChatGPT	1	249	Wiki
	PerplexityAI	2	149	Wiki
	ChatGPT	2	115	Wiki
Response	FELM	—	184	Google
	BINGCHAT	—	396	Google

Table 4: Evaluation datasets by label granularity, atomicity, and size. Abbreviations: KS = Knowledge Source; Wiki = Wikipedia.

C Annotation Examples

Table 6 shows annotated examples of decompositions produced by DAD.

D Comparative Analysis with Existing Literature

E Comparative Analysis with Existing Literature

Table 7 provides a side-by-side comparison of our method against relevant prior works, summarizing key differences and improvements.

F Error Analysis

Our work aims to enhance the decomposer for better verification accuracy. However, as our approach integrates multiple components, overall performance remains constrained by the limitations of retrieval and verification modules, which can still introduce errors despite improved decomposition. This is especially pronounced in open-world retrieval scenarios, such as web search, where temporal discrepancies in evidence—particularly for time-sensitive claims—can significantly impact results. To clarify the approach's limitations, we conducted an error analysis of misclassified claims, identifying whether failures arose from decomposition quality, evidence insufficiency, annotation ambiguity, or verifier reasoning. Appendix Table 8 shows representative error cases, illustrating the types of mismatches observed and their causes.

One of the recurring patterns from the decomposition perspective was the model's tendency to oversupport composite claims, in cases where claim decomposition led the model to treat complex, multipart statements as a single unit, resulting in support based on partial evidence (Table 8, example 1). Annotation ambiguity contributed to some mismatches, especially in cases where the ground truth label was debatable (Table 8, example 1). Verifier can lack world knowledge or overgeneralize. Verifier sometimes accepted ambiguous or outdated evidence, failing to critically assess its relevance or recency (Table 8, examples 2,3).

G Reasoning Traces Analysis

Following the error analysis, we qualitatively examine how the sequential decomposition prompt influences intermediate reasoning and final outputs, focusing on the four-step process described in Section 3: Step 1—Claim detection ($S_i \rightarrow$ *VerifiableSentence*); Step 2—Decontextualization (*VerifiableSentence* $\rightarrow$ *DecontextualizedSentence*); Step 3—Relationship identification (*DecontextualizedSentence* $\rightarrow$ *Relationships*); and Step 4—Claim extraction (*DecontextualizedSentence* + *Relationships* $\rightarrow$ *Claims*). Appendix Table 9 presents representative cases illustrating the contribution of each step.

In Example 1, Step 1 correctly excludes a lack-of-information statement ("but little else is known about her current whereabouts or status") and retains only the verifiable proposition. In Example 2, Step 2 confirms that all pronouns (e.g., "he") are resolved to explicit entity names, yielding a

Desiderata	3-way (%)	Fleiss' κ	Pairwise (%)	κ	r
Completeness	80.8	0.5	A1-A3: 85.0	0.5	0.5
			A1-A2: 85.9	0.5	0.5
			A3-A2: 90.9	0.6	0.6
Uniqueness	92.9	0.9	A1-A3: 94.0	0.8	0.8
			A1-A2: 93.9	0.8	0.8
			A3-A2: 97.0	0.9	0.9
Verifiability	81.9	0.7	A1-A3: 90.9	0.8	0.8
			A1-A2: 86.3	0.6	0.6
			A3-A2: 86.3	0.6	0.6
Coherence	73.6	0.5	A1-A3: 81.1	0.5	0.5
			A1-A2: 85.3	0.6	0.6
			A3-A2: 80.9	0.5	0.5
Clarity	81.2	0.7	A1-A3: 88.3	0.7	0.7
			A1-A2: 85.6	0.7	0.7
			A3-A2: 88.0	0.7	0.7

Table 5: Inter-annotator agreement statistics across the five desiderata. 3-way agreement shows percentage agreement among all three annotators (A1, A2, A3) with Fleiss' κ for multi-annotator reliability. Pairwise statistics include percentage agreement, Cohen's κ, and Pearson correlation coefficient (r) for each annotator pair.

self-contained *DecontextualizedSentence*; in this instance, the target entity was already explicit in the *VerifiableSentence* produced by Step 1. Step 3 is effective in both examples: in Example 1, it identifies temporal and expansion relations; in Example 2, it identifies temporal and membership/part-of relations. These intermediate annotations are crucial for Step 4, which splits the sentence into minimal, fully decontextualized factual units while preserving identified relationships. In both cases, the sentence-level label assigned by the verifier matches the ground-truth annotation.

H Implementation Details

Sentence Segmentation. We perform sentence segmentation using `spaCy`[7] to split model responses into individual sentences. For each target sentence, we construct a local context window including the $p = 2$ preceding and $f = 2$ following sentences, truncated to document boundaries as needed. This segmented structure forms the input tuples (Q, C_i, S_i) used for decomposition.

Synthetic Data Generation. We generate synthetic decompositions using Llama-3.1-405B-Instruct via Amazon Bedrock with temperature 0.35 and top-p 0.9. For each of the 15.7k sampled sentences, we construct a prompt containing the original question, surrounding context (preceding and following sentences), and the target sentence, then generate decompositions using our decomposition prompt shown in Appendix K.

SFT Warmup. We fine-tune Llama-3.1-8B-Instruct using LoRA (Hu et al., 2022) with rank $r = 32$ and $\alpha = 64$, targeting all attention and MLP projection layers. We train for 1 epoch on 13.7k examples with batch size 32 (8 per device, 4 gradient accumulation steps), learning rate 2×10^{-4} with cosine decay and 10% warmup, and weight decay 0.01. We use AdamW 8-bit optimizer and mixed precision training. Training uses the Llama-3.1 chat template with standard roles and applies loss only on assistant tokens. We train on a single A100 80GB GPU for approximately 8 hours using the Unsloth implementation (Daniel Han and team, 2023)[8].

RL Implementation. We fine-tune the SFT warm-up policy (trained for one epoch) using the Dr.GRPO implementation (Liu et al., 2025b). Training runs for 600 steps with a per-GPU batch size of 4 and gradient accumulation of 8, yielding an effective batch of 256 prompts per update. Each prompt generates 8 completions per update group. Prompts are truncated to 2,048 tokens and completions to 3,072 tokens within a 4,096-token context window. Optimization follows a constant-with-warmup schedule (20 warmup steps) with a learning rate of 2×10^{-6}. For memory

[7] https://spacy.io/

[8] https://github.com/unslothai/unsloth

efficiency, LoRA (Hu et al., 2022) adapters with rank r=16 and α=32 are applied to all attention and MLP projection layers. The policy is implemented and trained using the Verifiers implementation (Brown, 2025) [9]. Training is conducted on an 8×A100 (40GB) setup and completes in approximately 15 hours.

I Reward Implementation Details

We provide the implementation details of our reward terms in this section.

I.1 Checklist Reward

- ☐ **Complete Verifiable (Hard Gate):** Is the subclaim a single complete sentence that makes one verifiable proposition?
- ☐ **Retrieval Relevant (0.40):** Can a reader without access to the original question or context use this subclaim alone to retrieve correct, relevant evidence?
- ☐ **Qualifiers Sufficient (0.15):** If the subclaim depends on qualifiers (time, attribution, conditions, scope, location, etc.), are they present?
- ☐ **References Explicit (0.30):** Are entities, events, and relations explicitly named and unambiguous?
- ☐ **No Ungrounded Additions (0.15):** Does the subclaim avoid adding information not present in the question, context, or sentence?

We compute the checklist reward using Llama-3.3-70B-Instruct via Amazon Bedrock (temperature 0.2, top-p 0.9). For each subclaim, the judge outputs `Yes`, `No`, or `NA` for each criterion. We apply a hard constraint: if `complete_verifiable` $\neq$ `Yes`, the subclaim receives zero reward. Otherwise, we compute a weighted average over the remaining non-`NA` criteria using weights: 0.40 (retrieval relevance), 0.30 (explicit references), 0.15 (qualifiers), and 0.15 (no ungrounded content). The final reward for a completion is the geometric mean of all subclaim scores, with invalid or malformed judge outputs assigned zero reward. The complete prompt is in Appendix K.

[9] https://github.com/PrimeIntellect-ai/verifiers

I.2 Format Reward

- ☐ **Tags Present (0.40):** Both `<think>...</think>` and `<output>...</output>` appear.
- ☐ **Order & Clean (0.10):** The completion contains only these two blocks in order, with nothing extraneous.
- ☐ **List Parsed (0.40):** The `<output>` parses as a list of strings (strict JSON or lenient literal).
- ☐ **List Quality (0.10):** All items are non-empty strings; at least one item is required.

The format reward uses regex-based parsing to verify: (i) presence of `<think>` and `<output>` tags, (ii) correct tag ordering with no extraneous content, (iii) successful parsing of `<output>` as a list, and (iv) list validity (non-empty, no blank strings). We combine these as a weighted sum with weights 0.40 (tags present), 0.10 (ordering), 0.40 (parseable list), and 0.10 (valid list).

J Use of AI Assistants

Portions of this manuscript benefited from the use of an AI assistant for language refinement. All editorial decisions and content oversight remained the sole responsibility of the authors.

Question	Businessman, a Jewish American, grew up and lived in New York City, associated with the Democratic Party, and eventually became very famous. His name is "Bernie"-what?				
Sentence	Bernie Sanders.				
Subclaims	Bernie Sanders				
	Completeness	**Redundancy**	**Verifiability**	**Coherence**	**Clarity**
Human Annotation Scores	Majority 0	Majority 1	Average 0	Average 1	Average 0

Question	Tell me a bio of Paul Kruger.				
Sentence	Kruger was a prominent figure in the Boer War (1899-1902), leading the Boers in their defense against the British Empire. Kruger was a staunch believer in Boer independence and was instrumental in securing the Transvaal's independence from British rule in 1881.				
Subclaims	1. Paul Kruger was a prominent figure in the Boer War (1899-1902). 2. Paul Kruger led the Boers in their defense against the British Empire during the Boer War. 3. Paul Kruger was a staunch believer in Boer independence. 4. Paul Kruger was instrumental in securing the Transvaal's independence from British rule in 1881.				
	Completeness	**Redundancy**	**Verifiability**	**Coherence**	**Clarity**
Human Annotation Scores	Majority 1	Majority 1	Average 1	Average 1	Average 1

Question	Tell me a bio of Rakhi Sawant.				
Sentence	Rakhi Sawant established herself as a sex symbol in Bollywood. She was a contestant on the first season of the Indian reality television series Bigg Boss 1 (2006) and a challenger and finalist in Bigg Boss 14 (2020).				
Subclaims	1. Rakhi Sawant established herself as a sex symbol in the Indian film industry. 2. Rakhi Sawant was a contestant on the first season of the Indian reality television series Bigg Boss 1 (2006). 3. Rakhi Sawant was a challenger and finalist in Bigg Boss 14 (2020).				
	Completeness	**Redundancy**	**Verifiability**	**Coherence**	**Clarity**
Human Annotation Scores	Majority 0	Majority 1	Average 1	Average 0.67	Average 1

Table 6: Subclaims and Human Annotation Scores for Example Questions

Method	What is Similar to Our Approach	What is Different / Missing Compared to Ours
FActScore (Min et al., 2023)	Focuses on atomic fact extraction; uses selection and extraction steps.	Over-decomposes (too many subclaims); does not optimize for verifier alignment; lacks explicit decontextualization and relationship identification.
VeriScore (Song et al., 2024)	Emphasizes verifiability in claim extraction; uses prompt-based decomposition.	Does not explicitly decontextualize claims; lacks relationship identification; does not jointly optimize decomposition and verifier alignment.
Molecular Facts (Gunjal and Durrett, 2024)	Targets fully decontextualized, minimal subclaims.	Focuses only on decontextualization; does not address selection, relationship identification, or verifier alignment.
Claimify (Metropolitansky and Larson, 2025)	Separates selection, disambiguation, and extraction into distinct steps.	Uses complex, multi-stage process with closed-source models; does not jointly optimize decomposition and verifier alignment; lacks efficiency of single model call.
DyDecomp (Lu et al., 2025)	RL-based optimization for when to decompose, using verifier feedback.	Only decides when to decompose, not how; does not improve decomposition quality or address relationships and decontextualization.
DndScore (Wanner et al., 2024b)	Focuses on decontextualization and decomposition for factuality verification.	Does not integrate selection, relationship identification, and verifier alignment in a unified framework.
VeriFastScore (Rajendhran et al., 2025b)	Aims for efficient long-form factuality evaluation with refined fact extraction.	Prioritizes efficiency; does not address relationship identification or joint optimization with verifier.

Table 7: Comparison of Decomposer-Focused Methods: Similarities and Differences with Our Approach

Field	Content
Example 1: Frederick Howard Taylor	
Question	Tell me a bio of Frederick Howard Taylor.
Original Claim	Howard Taylor attended the Royal College of Surgeons where he received three high honors in his postgraduate studies.
Subclaims	1. Howard Taylor attended the Royal College of Surgeons where he received three high honors in his postgraduate studies.
Ground Truth Label	Not Supported (NS)
Model Prediction	Supported (S)
Evidence	Taylor was a Fellow of the Royal College of Surgeons and received three high honors.
Error Cause	Decomposer could have decomposed the claim further. Verifier conflated "fellowship" with "attendance," supporting the claim without direct evidence of attendance. This reflects a reasoning failure and insufficiently strict evidence matching.
Example 2: Mauro Icardi	
Question	Tell me a bio of Mauro Icardi.
Original Claim	He plays as a striker for Süper Lig club Galatasaray, on loan from Ligue 1 club Paris Saint-Germain and the Argentina national team.
Subclaims	1. Mauro Icardi plays as a striker for Galatasaray. 2. Mauro Icardi is on loan from Paris Saint-Germain. 3. Mauro Icardi is on loan from the Argentina national team.
Ground Truth Label	Not Supported (NS)
Model Prediction	Supported (S)
Evidence	The evidence confirms Icardi's club status and loan from PSG, but there is no such thing as a loan from a national team in football.
Error Cause	Lack of world knowledge in the verifier led to its failure to recognize the impossibility of a national team loan.
Example 3: Eric Hacker	
Question	Tell me a bio of Eric Hacker.
Original Claim	As of March 2023, Eric Hacker is a free agent.
Subclaims	1. Eric Hacker is a free agent as of March 2023.
Ground Truth Label	Not Supported (NS)
Model Prediction	Supported (S)
Evidence	Eric Lynn Hacker (born March 26, 1983) is an American professional baseball pitcher who is a free agent. Several mentions of him being a free agent in earlier periods. However, no mention of March 2023.
Error Cause	Verifier overgeneralized or relied on outdated information, supporting the claim without up-to-date evidence.

Table 8: Detailed Verification Error Analysis with Subclaims and Evidence

Field	Content
Example 1: Lina Medina [Dataset: ChatGPT, Atomicity 1]	
Question	Tell me a bio of Lina Medina.
Original Claim	Lina lived a relatively private life after the birth of her son and later married in 1970, but little else is known about her current whereabouts or status.
Reasoning	STEP 1: The target sentence contains verifiable claims about Lina Medina's life after the birth of her son, specifically that she lived a relatively private life and later married in 1970. However, the sentence also mentions that little else is known about her current whereabouts or status, which is a lack-of-information statement and should be excluded. The verifiable sentence is: "Lina lived a relatively private life after the birth of her son and later married in 1970." STEP 2: To decontextualize the sentence, we need to ensure it is clear, self-contained, and contextually relevant. The sentence already mentions Lina's name, and the context provides the necessary information about her son's birth. The decontextualized sentence is: "Lina Medina lived a relatively private life after the birth of her son and later married in 1970." STEP 3: Analyzing the decontextualized sentence, we can identify the following relationships: - Temporal relationship: Lina lived a private life after the birth of her son and later married in 1970 (time sequence). - Expansion relationship: The sentence provides additional information about Lina's life after the birth of her son (elaboration). STEP 4: To split the claim into the smallest possible factual units, we can break it down into two separate claims: - Claim 1: Lina Medina lived a relatively private life after the birth of her son. - Claim 2: Lina Medina married in 1970.
Subclaims	1. Lina Medina lived a relatively private life after the birth of her son. 2. Lina Medina married in 1970.
Model prediction	Not Supported (NS)
Ground truth	Not Supported (NS)
Example 2: Ronaldo (Brazilian footballer) [Dataset: PerplexityAI, Atomicity 1]	
Question	Tell me a bio of Ronaldo (Brazilian footballer).
Original Claim	At age 17, he was the youngest member of the Brazilian squad that won the 1994 FIFA World Cup.
Reasoning	STEP 1: The target sentence contains a specific and verifiable proposition about Ronaldo's age and his participation in the 1994 FIFA World Cup. This meets the criteria for a verifiable claim. VerifiableSentence: At age 17, Ronaldo was the youngest member of the Brazilian squad that won the 1994 FIFA World Cup. STEP 2: To decontextualize the sentence, we need to ensure it stands alone as a query-relevant, unambiguous statement. The sentence already mentions Ronaldo by name and specifies the event (1994 FIFA World Cup) and his role (youngest member of the Brazilian squad). No additional information from the Question or Context is needed to resolve ambiguity. DecontextualizedSentence: At age 17, Ronaldo was the youngest member of the Brazilian squad that won the 1994 FIFA World Cup. STEP 3: Identify relationships in the sentence. There is a temporal relationship indicated by "At age 17," which specifies Ronaldo's age at the time of the event. There is also a membership/part-of relationship, as Ronaldo is part of the Brazilian squad. Relationships: Temporal, Membership/Part-of. STEP 4: Split the claim into the smallest possible factual units. From the DecontextualizedSentence and identified relationships, we can split the claim into two specific, verifiable, and fully decontextualized claims. Claims: 1. Ronaldo was 17 years old when the Brazilian squad won the 1994 FIFA World Cup. 2. Ronaldo was the youngest member of the Brazilian squad that won the 1994 FIFA World Cup.
Subclaims	1. Ronaldo was 17 years old when the Brazilian squad won the 1994 FIFA World Cup. 2. Ronaldo was the youngest member of the Brazilian squad that won the 1994 FIFA World Cup.
Model prediction	Supported (S)
Ground truth	Supported (S)

Table 9: Decomposition analysis with reasoning traces. Highlighted segments are discussed in Appendix G.

Field	Content
Example: Harrison Ford	
Question	Tell me a bio of Harrison Ford.
Text	He was born on July 13, 1942, in Chicago, Illinois, to former radio actress Dorothy and advertising executive John William "Christopher" Ford.
Label	Supported
FactScore (FS) Subclaims	['He was born on July 13, 1942.', 'He was born in 1942.', '1942 is a year.', 'July 13 is a date.', 'July 13 is a date in 1942.', 'He was born on July 13.', 'He was born in July 1942.', 'July is a month.', 'July is a month of 1942.', 'He was born in July.', 'He was born in July 1942.', 'He was born in 1942.', <.. repeated variants of 'He was born in the year of his birth, 1942.' ..>]
Prediction using FS subclaims	Not Supported
VeriScore (VS) Subclaims	['Harrison Ford was born on July 13, 1942.', 'Harrison Ford was born in Chicago, Illinois.', 'Harrison Ford's father was a former radio actress named Dorothy.', 'Harrison Ford's father was an advertising executive named John William "Christopher" Ford.']
Prediction using VS subclaims	Not Supported
DAD Subclaims	['Harrison Ford was born on July 13, 1942.', 'Harrison Ford was born in Chicago, Illinois.', 'Harrison Ford's mother was a former radio actress named Dorothy.', 'Harrison Ford's father was an advertising executive named John William "Christopher" Ford.']
Prediction using DAD subclaims	Supported
DAD Reasoning Trace	<think> STEP 1: The target sentence contains specific and verifiable information about Harrison Ford's birth, including the date, place, and parents. VerifiableSentence: He was born on July 13, 1942, in Chicago, Illinois, to former radio actress Dorothy and advertising executive John William "Christopher" Ford. STEP 2: DecontextualizedSentence: Harrison Ford was born on July 13, 1942, in Chicago, Illinois, to former radio actress Dorothy and advertising executive John William "Christopher" Ford. STEP 3: Relationships: Membership/Part-of (Harrison Ford's birth family). STEP 4: Claims: 1. Harrison Ford was born on July 13, 1942. 2. Harrison Ford was born in Chicago, Illinois. 3. Harrison Ford's mother was a former radio actress named Dorothy. 4. Harrison Ford's father was an advertising executive named John William "Christopher" Ford. </think> <output>["Harrison Ford was born on July 13, 1942.", "Harrison Ford was born in Chicago, Illinois.", "Harrison Ford's mother was a former radio actress named Dorothy.", "Harrison Ford's father was an advertising executive named John William C̈hristopherF̈ord."]</output>

Table 10: Examples comparing decompositions across several methods. Highlighted issues with decompositions.

Field	Content
Example: Scientist (musician)	
Question	Tell me a bio of Scientist (musician).
Text	He was a protégé of King Tubby and worked alongside other pioneers of the genre such as Bunny Lee, Philip Smart, Pat Kelly, and Prince Jammy.
Label	Supported
FactScore (FS) Subclaims	['He was a protégé of King Tubby.', 'King Tubby is a person.', 'He was a protégé.', 'He was a protégé of King Tubby.', 'King Tubby is a person who was a mentor.', 'He worked alongside other pioneers of the genre.', 'The genre is a type of music.', 'Bunny Lee is a pioneer of the genre.', 'He worked alongside Bunny Lee.', 'Philip Smart is a pioneer of the genre.', 'He worked alongside Philip Smart.', 'Pat Kelly is a pioneer of the genre.', 'He worked alongside Pat Kelly.', 'Prince Jammy is a pioneer of the genre.', 'He worked alongside Prince Jammy.']
Prediction using FS subclaims	Not Supported
VeriScore (VS) Subclaims	['Hopeton Overton Brown, known as Scientist, was a protégé of King Tubby.', 'Hopeton Overton Brown, known as Scientist, worked alongside King Tubby.', 'Hopeton Overton Brown, known as Scientist, worked alongside Bunny Lee.', 'Hopeton Overton Brown, known as Scientist, worked alongside Philip Smart.', 'Hopeton Overton Brown, known as Scientist, worked alongside Pat Kelly.', 'Hopeton Overton Brown, known as Scientist, worked alongside Prince Jammy.']
Prediction using VS subclaims	Not Supported
DAD Subclaims	['Hopeton Overton Brown (Scientist) was a protégé of King Tubby.', 'Hopeton Overton Brown (Scientist) worked alongside other pioneers of the genre such as Bunny Lee, Philip Smart, Pat Kelly, and Prince Jammy.']
Prediction using DAD subclaims	Supported
DAD Reasoning Trace	<think> STEP 1: VerifiableSentence: He was a protégé of King Tubby and worked alongside other pioneers of the genre such as Bunny Lee, Philip Smart, Pat Kelly, and Prince Jammy. STEP 2: DecontextualizedSentence: Hopeton Overton Brown (Scientist) was a protégé of King Tubby and worked alongside other pioneers of the genre such as Bunny Lee, Philip Smart, Pat Kelly, and Prince Jammy. STEP 3: Relationships: Attribution; Membership/Part-of. STEP 4: Claims: - Hopeton Overton Brown (Scientist) was a protégé of King Tubby. - Hopeton Overton Brown (Scientist) worked alongside other pioneers of the genre such as Bunny Lee, Philip Smart, Pat Kelly, and Prince Jammy. </think> <output>["Hopeton Overton Brown (Scientist) was a protégé of King Tubby.", "Hopeton Overton Brown (Scientist) worked alongside other pioneers of the genre such as Bunny Lee, Philip Smart, Pat Kelly, and Prince Jammy."]</output>

Table 10: Examples comparing decompositions across several methods. Highlighted issues with decompositions. (continued)

K Prompts Used in Experiments

In this section, we include all prompts used in our experiments.

K.1 Decomposition System Prompt

Decomposition Prompt

You are an assistant for a group of fact-checkers. You will be given a question inside the <question></question> tags, which was asked about a source text (it may also be referred to by other names, e.g., a dataset). You will also be given an excerpt from a response to the question inside the <context></context> tags. If the context contains "[...]", this means you are **not** seeing the full response. You will also be given a target sentence from the response inside the <sentence></sentence> tags.
Your task is to identify all specific and verifiable propositions in the target sentence and ensure each proposition is **decontextualized** while maintaining any relationships between the claims. To do this, you must reason step by step wrapped inside <think>...</think> using the steps provided. The final results must always be output inside <output>...</output>. Follow the rules strictly.
Follow these rules through the steps:

- A **verifiable proposition/claim** is a concrete, checkable assertion (event, state, relation, attribution) whose truth can be assessed against reliable external knowledge (e.g., Wikipedia or Google Search). The **truth** or **relevance** of the claim does **not** affect inclusion.
- Use the **Question/Context only** to resolve references (pronouns, definite phrases, acronyms), not external knowledge.
- Treat **attributions** as verifiable facts about speech/stance ("X says/claims/reports/ predicts Y"), even if Y is speculative.
- **Include as verifiable**: entity/role facts; events (who/what/when/where); quantified facts (numbers/dates/comparisons); attributed statements/stances; context-bridged answers (e.g., Q: "Who is CEO?" → A: "John.").
- **Exclude**: lack-of-information statements; introductions/previews; pure conclusions/summaries; generic/normative opinions; unattributed speculation/hypotheticals; instructions/suggestions; headings/teasers.
- When a sentence mixes generic language with a specific fact, extract the **specific fact only**.
- **Decontextualized claims** must be self-contained, preserve the original meaning relative to context, and refer to entities by **name** (avoid pronouns). Include **time/place** when present. Keep each claim to one sentence with zero or at most one embedded clause.
- If Context contains "[...]", you may **not** assume unseen material.

Follow these reasoning steps inside <think></think>:
STEP 1 — Decide if the sentence has verifiable claims (TargetSentence → VerifiableSentence)
Input: Target sentence contained in <sentence></sentence>.
Determine whether this particular sentence contains at least one specific and

verifiable proposition, and if so, return a complete sentence that only contains verifiable information.
Apply Include/Exclude rules strictly.
If YES → produce VerifiableSentence with the verifiable information.
If NO → stop and final output is <output>No verifiable claim</output>.

STEP 2 — Rewrite the sentence so it is clear and self-contained (VerifiableSentence → DecontextualizedSentence)
Input: VerifiableSentence from Step 1.
Replace pronouns, partial names, and undefined acronyms/abbreviations only if the full form appears in the Question or the Context.
If the sentence, when read in isolation, could have more than one meaning, use the Question/Context to select the correct interpretation so that the sentence can stand alone.
If ambiguity remains and multiple reasonable meanings cannot be resolved using the Question/Context, stop and output: <output>Cannot be decontextualized</output>.
Otherwise, produce: DecontextualizedSentence: a single, self-contained sentence that preserves the original meaning while resolving references and ambiguity using only the given Question/Context.

STEP 3 — Find relationships between the parts of the claim (DecontextualizedSentence → Relationships)
Input: DecontextualizedSentence from Step 2. Identify relationships in the sentence, if present:

- **Attribution** (e.g., "X says Y.")
- **Causal** (cause, effect, purpose, condition; e.g., "X happened because Y." / "If X, then Y.")
- **Temporal** (time sequence or overlap; e.g., "X happened after Y." / "X happened while Y occurred.")
- **Comparison** (contrast, concession, similarity; e.g., "X increased, while Y decreased." / "Even though X, Y happened.")
- **Expansion** (elaboration, example, detail, alternative; e.g., "X, such as Y." / "Instead of X, Y.")
- **Negation** (explicit negatives; e.g., "X did not happen.")
- **Membership/Part-of** (subset or composition; e.g., "X is part of Y." / "X is a member of Y.")

Produce a list of identified relationships (or "No relationships").

STEP 4 — Split the claim into the smallest possible factual units (DecontextualizedSentence + Relationships → Claims)
Input: DecontextualizedSentence from Step 2 and Relationships from Step 3.
Split the DecontextualizedSentence into **specific, verifiable, and fully decontextualized claims**, meaning it can be understood in isolation (i.e., without the question, the context, and the other propositions), and its meaning in

isolation matches its meaning when interpreted alongside the question, the context, and the other propositions.
Preserve all **numbers, time, place, comparisons, negations, attributions**.
Do not invent details beyond the given text.
Quotations may be extracted verbatim but must be connected via **Attribution**.
Final output: <output>[claim1, claim2, ...]</output>.

Format the response as:

<think>
[Step 1: Selection reasoning → VerifiableSentence or stop
Step 2: Decontextualization reasoning → DecontextualizedSentence or stop
Step 3: Relationship identification reasoning → Relationships
Step 4: Decomposition reasoning → Claims]
</think>
<output>
[Final result: either "No verifiable claim", "Cannot be decontextualized",
or list of decomposed claims from Step 4]
</output>

K.2 Decomposition Checklist Prompt

Decomposition Checklist Prompt

You are an assistant for a group of fact-checkers. Your job is to help verify whether the extracted subclaims from a previous claim extraction task satisfy various requirements for being decontextualized—i.e., whether they stand alone and correctly encode the relationship asked by the checklist—using **ONLY** the materials provided. Do not use external knowledge. Do not approximate.
INPUTS
You will be given:
<question>the user's question that generated the model's response</question
<context>a portion of the model's response surrounding the target sentence (including some preceding and following text)</context
<sentence>the specific sentence from the context to be decomposed</sentence
<subclaims>["subclaim 1", "subclaim 2"]</subclaims>
NOTES

- Evaluate each subclaim independently; do not compare across subclaims.
- The <context> may be partial; if it contains [...] you are not seeing the full response.
- Judge only using information in <question>, <context>, <sentence>, and <subclaims>. Do not use outside knowledge.
- If something cannot be determined from the provided materials, mark the check as NA.
- Only checklist items **3** (qualifiers_sufficient) and **4** (references_explicit) may return NA.

CHECKLIST
For each subclaim, answer these questions with Yes/No/NA. Use the JSON keys exactly as shown.

1. **complete_verifiable**
   Is the subclaim a single complete sentence that makes one verifiable proposition?
   A verifiable proposition means a statement that can be objectively verified as true or false based on empirical evidence or reality (not an opinion, vague description, or evaluative language).

2. **retrieval_relevant**
   Can a reader with no access to the original question or context use this subclaim alone to retrieve the correct and relevant evidence for the sentence?
   Yes: The subclaim is stand-alone and disambiguating—it explicitly names the target entity/event and includes the necessary salient identifiers from the provided materials (e.g., roles/titles, aliases, timeframe, locations, distinctive actions). It does not introduce unsupported attributes.
   No: It is background that cannot be linked to the question, is off-topic, or too vague.

3. **qualifiers_sufficient**
   If the subclaim depends on qualifiers (time, attribution, conditions, scope, location, etc.) in order to be a complete and accurate statement, are those qualifiers present in the subclaim?
   If no qualifier is necessary for the subclaim to stand as a complete, faithful proposition, mark Yes.
   If it is unclear from the provided materials whether a qualifier is necessary, mark NA.

4. **references_explicit**
   Are all entities, events, and relations in the subclaim explicitly named and unambiguous, using only the information available in <question>, <context>, or <sentence>?
   Names must be fully resolved: if both a first and last name (or equivalent full form) are present in the materials, use the full form (e.g., “John Smith” instead of just “John” or “Smith”).
   Pronouns are never acceptable: if the subclaim uses he, she, they, it, this, that, these, those, former, latter, etc. in place of an entity or event → mark No.
   Acronyms must be expanded if the expansion is given in the materials; otherwise mark No.
   If ambiguity cannot be resolved from the provided materials → NA. Otherwise → Yes.

5. **no_ungrounded_additions**
   Does the subclaim avoid adding information not explicitly present in the question, context, or sentence?

OUTPUT FORMAT
Output **STRICT JSON only** (a top-level list of objects). Each object must include both "checks" and "rationales" (single-sentence, grounded in the provided materials, no external info).
Schema: {[
  {"id": "index",
    "checks": {"complete_verifiable": "Yes|No", "retrieval_relevant": "Yes|No", "qualifiers_sufficient": "Yes|No|NA", "references_explicit": "Yes|No|NA",

```
"no_ungrounded_additions": "Yes|No"},
    "rationales": {"complete_verifiable": "One concise sentence explaining the decision.", "retrieval_relevant": "One concise sentence explaining the decision.", "qualifiers_sufficient": "One concise sentence explaining the decision.", "references_explicit": "One concise sentence explaining the decision.", "no_ungrounded_additions": "One concise sentence explaining the decision."}
  }] }
```

Here is your input:

```
<question>{QUESTION}</question>
<context>{CONTEXT}</context>
<sentence>{SENTENCE}</sentence>
<subclaims>{SUBCLAIMS_JSON_ARRAY}</subclaims>
```