SAFE-FEC: Semantically Constrained Adversarial Data Construction for Factual Error Correction

Xiaobao Wang[1] **Lei Zhu**[1] **Jianbiao Yang**[2]
Chenyang Wang[3] **Longbiao Wang**[1*] **Jianwu Dang**[1,4]

[1]Tianjin Key Laboratory of Cognitive Computing and Application,
School of Artificial Intelligence, Tianjin University, Tianjin, China
[2]College of Management and Economics, Tianjin University, Tianjin, China
[3]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China
[4]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

{wangxiaobao,maplesakura,Jianbiao,longbiao_wang}@tju.edu.cn
chenyangwang@ieee.org jw.dang@siat.ac.cn

Abstract

Factual Error Correction (FEC) aims to revise factually incorrect claims into evidence-supported statements, relying on high-quality correction data for effective training and evaluation. However, existing error-injection methods largely rely on predefined corruption strategies without considering the capabilities of current correctors, often producing errors that are easily repaired. Simply increasing error complexity is insufficient, as unconstrained mutations may compromise correction validity. To bridge this gap, we propose SAFE-FEC, a semantically constrained adversarial framework for correction-aware FEC data construction. SAFE-FEC explores diverse candidate errors through multi-perspective generation, filters semantic drift, and uses corrector feedback to refine candidates toward more challenging correction instances. Experiments on three datasets demonstrate that SAFE-FEC consistently generates more challenging errors for both LLM-based and FEC-specific correctors. On FECDATA, the average LLM-judge correction score decreases by 36.9% compared with human-written examples and by 53.4% compared with CECoR-generated examples. In a human audit, 98% of examples contain a factual error and 100% remain semantically anchored.

1 Introduction

Factual Error Correction (FEC) aims to revise factually incorrect claims into evidence-supported statements, playing an important role in improving the reliability of Large Language Models (LLMs) [1, 2]. Unlike factual error detection, FEC requires not only identifying factual inconsistencies but also correcting them according to the given evidence. Developing effective FEC systems therefore relies on high-quality correction data consisting of erroneous claims, their corresponding corrections, and supporting evidence. However, constructing such data is challenging. Manual annotation is costly and difficult to scale, motivating automatic FEC data construction. Early approaches follow the *mask-then-correct* paradigm, which masks potentially erroneous spans in false claims and reconstructs correct statements using evidence [3, 4]. However, their effectiveness depends heavily on accurate error localization. To overcome this limitation, recent error-injection methods, such as LIFE [5] and CECoR [6], generate false claims by introducing factual errors into evidence-supported statements. This strategy directly produces correction pairs without requiring error localization.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

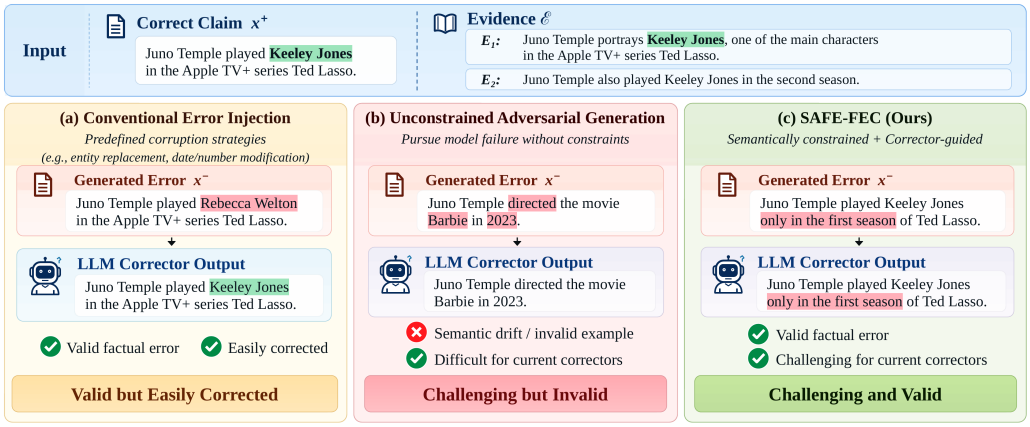

Figure 1: Illustrative comparison of conventional error injection, unconstrained adversarial generation, and SAFE-FEC.

Despite their effectiveness, existing error-injection methods largely rely on predefined corruption strategies without considering the correction capabilities of current models. As illustrated in Fig. 1(a), conventional error injection typically introduces straightforward factual modifications, such as incorrect entities, dates, or numbers. Although these modifications produce valid factual errors, strong LLM correctors can often repair them through direct evidence matching. More importantly, the apparent complexity of a factual error does not necessarily reflect its correction difficulty. An error involving multiple facts may still be easily corrected, while a seemingly simple modification may expose a corrector's reasoning limitations. Therefore, the difficulty of generated errors should be assessed against the actual correction capabilities of current models, rather than determined solely by predefined corruption rules.

Nevertheless, simply maximizing correction difficulty is insufficient. As shown in Fig. 1(b), unconstrained adversarial generation may produce errors that are difficult for correctors but deviate from the original claim's semantic context. Such mutations may introduce semantic drift, unverifiable information, or ambiguous correction targets, making the resulting examples difficult for reasons unrelated to factual reasoning. Consequently, a corrector's failure on these examples does not necessarily indicate a limitation in its factual correction capability. This reveals a fundamental tension in FEC data construction: generated errors should be sufficiently challenging to expose the weaknesses of current correctors, while remaining semantically anchored to the original claim with an identifiable correction target. This raises a key question: *How can we adaptively generate difficult correction instances while controlling semantic drift and assessing their validity?*

To address this challenge, we propose SAFE-FEC, a semantically constrained adversarial framework for correction-aware FEC data construction. As illustrated in Fig. 1(c), SAFE-FEC aims to generate factual errors that are both challenging for current correctors and semantically consistent with the original claims. Specifically, SAFE-FEC explores diverse candidate errors through multi-perspective generation and cross-perspective refinement, while a semantic guard filters candidates that deviate from the original semantic context. More importantly, rather than relying solely on predefined corruption strategies, SAFE-FEC incorporates feedback from current correctors to adaptively refine error generation, progressively identifying challenging correction instances. Experiments on three benchmark datasets demonstrate that SAFE-FEC consistently produces more challenging FEC data for both LLM-based and FEC-specific correctors. A human audit further assesses factual status, evidence sufficiency, correction identifiability, semantic anchoring, and fluency. Our main contributions are summarized as follows:

- We propose SAFE-FEC, a correction-aware framework that shifts FEC data construction from predefined error injection to feedback-driven adversarial exploration.
- We integrate semantic anchoring with corrector-guided refinement to generate challenging candidate errors and assess their validity through a human audit.

- Extensive experiments demonstrate that SAFE-FEC exposes limitations of existing correctors, while a human audit evaluates the factual status, evidence sufficiency, correction identifiability, semantic anchoring, and fluency of the generated examples.

2 Related Work

2.1 Evidence-Grounded Factual Error Correction

Factual error correction (FEC) extends factual verification from predicting a support/refute label to revising a claim into an evidence-supported statement [4]. Unlike verification, FEC requires a structured repair: the system must preserve correct information, identify the factual inconsistency, and produce a minimally edited correction. Because manually annotated triples of erroneous claims, evidence, and corrected claims are costly, early FEC work derived supervision from fact verification datasets. A representative paradigm is mask-then-correct, where a masker identifies potentially erroneous spans in a false claim and a corrector fills the masked positions using evidence [4]. Subsequent work improves correction through different mechanisms. VENCE formulates FEC as iterative constrained editing guided by an offline fact-verification score [7], while FLEEK builds an end-to-end pipeline for claim extraction, evidence retrieval, factuality evaluation, and revision suggestion [8]. These methods focus primarily on how to perform factual repair. In contrast, our work focuses on how to construct FEC instances that are not only valid and evidence-grounded, but also diagnostically challenging for modern correctors.

2.2 Synthetic Supervision and Factual Error Injection

Synthetic supervision is central to FEC because paired correction data is scarce. However, mask-then-correct methods depend heavily on error localization: over-masking may erase valid information, incorrect masking may distort the correction target, and missed errors leave the corrector unaware of the key inconsistency. PivotFEC identifies these issues as major bottlenecks and introduces factual error injection as a pivot task for few-shot FEC [9]. A later line of work reverses the construction direction: instead of locating errors in false claims, it starts from correct claims and injects factual errors. LIFE proposes a mask-corrupt-correct framework that trains a corruptor to generate false/correct pairs for downstream correction [5]. CECoR further studies correction-oriented synthetic data construction for FEC and serves as a strong comparison method in our experiments [6]. These approaches show that error injection is useful for FEC, but they mainly optimize for producing plausible corruptions. SAFE-FEC instead asks whether a generated error is useful as a challenging correction example: it should be close enough to the source claim to define an identifiable correction target, yet difficult enough to challenge current evidence-based correctors.

2.3 Evidence-Grounded Verification Benchmarks

FEC commonly builds on fact verification resources because they provide claims and evidence annotations that can be repurposed for correction. FEVER introduced large-scale claim verification against Wikipedia evidence [3]. HOVER extends this setting to multi-hop verification over multiple Wikipedia articles [10], and FEVEROUS further broadens the evidence space by combining unstructured sentences with structured table cells [11]. While these datasets are primarily designed for verification, we use them as starting points for adversarial FEC construction. Given a supported claim and its evidence, SAFE-FEC generates an erroneous variant and treats the original claim as the correction target. HOVER and FEVEROUS are particularly useful for this purpose because their multi-hop and heterogeneous evidence structures support subtle relational, contextual, and evidence-composition errors that are difficult to capture with simple entity or date substitutions.

2.4 Adversarial Data Generation and LLM-Agent Optimization

Adversarial data generation and hard negative mining are widely used to expose model weaknesses [12, 13]. However, FEC imposes stricter constraints than generic adversarial generation. A model failure is not necessarily a useful FEC example: the failed input may be ambiguous, unnatural, semantically detached from the source claim, or lack a clear correction target. Such examples may reduce performance without meaningfully testing factual repair. SAFE-FEC differs from ordinary

adversarial generation in its success condition. It uses corrector failure only after the candidate remains semantically anchored to the original claim and forms a valid evidence-grounded correction problem. Thus, SAFE-FEC is not merely hard-negative mining; it searches for errors that are simultaneously plausible factual-error instances, identifiable correction instances, and stress tests of current correctors.

SAFE-FEC is also related to LLM-agent methods that use critique, refinement, debate, or LLM-as-a-judge evaluation [14–17]. SALF, for instance, uses natural-language feedback to iteratively improve fake-news generation and detection agents [18]. Our work shares the idea that language feedback can guide agent behavior, but differs in task and objective. SALF targets fake-news generation and binary detection, whereas SAFE-FEC targets evidence-grounded correction, where the output is a repaired claim. More importantly, SAFE-FEC binds generation, critique, and feedback to FEC-specific constraints: semantic anchoring, correction identifiability, evidence grounding, and correction resistance. Its contribution is therefore not an agent pipeline alone, but a correction-aware data construction mechanism tailored to factual error correction.

3 The Proposed Method

SAFE-FEC constructs challenging factual error correction (FEC) examples from a claim x^+ labeled as supported by its source dataset and the associated evidence E. It proposes a mutated claim x^- and records the instance (E, x^-, x^+), with x^+ as the intended correction. The pipeline in Fig. 2 comprises four stages: (1) *Multi-Perspective Generation & Refinement*, which generates candidates and regenerates them after cross-perspective critique and conditional prompt refinement; (2) *Selection & Composition*, which selects or combines refined candidates; (3) *Semantic Guard*, which checks anchoring to the source claim; and (4) *Corrector-in-the-Loop Test*, which tests resistance to recovery of the designated target. **Appendix** B summarizes the notation, Algorithm 1 specifies the implemented procedure, and Table 8 identifies the model used by each module.

3.1 Task and Operational Acceptance Criteria

An ideal instance has three properties: x^- conflicts with E while x^+ is supported by E; the mutation preserves the source entity and event so that x^+ is an identifiable repair target; and the mutation is difficult for current evidence-based correctors to repair. These properties serve different purposes. A correction failure alone can arise from an unverifiable addition, semantic drift, or a non-unique target rather than a challenging factual error.

The current implementation approximates these requirements with distinct checks. Evidence-conditioned generators and a judge instructed to assess evidence-checkability propose and select errors; a semantic guard checks whether the selected mutation stays anchored to x^+; and an adaptive corrector tests whether the intended correction can be recovered. The guard receives $(\tilde{x}_t, x^+)$, not E, so it does not itself verify factual incorrectness or evidence sufficiency. Nor does corrector failure establish either property. We assess those properties separately in the human audit (Appendix I) and discuss the remaining validity limitation in **Appendix** A.

3.2 Multi-Perspective Generation & Refinement

Candidate generation begins with four factual-error roles $\mathcal{K} = \{\text{fact}, \text{fiction}, \text{context}, \text{logic}\}$. The *fact* role targets local factual changes such as dates, names, entities, and attributes. The *fiction* role introduces hallucinated or fabricated content. The *context* role changes conditions, background, or scope. The *logic* role targets contradictions and relation-level inconsistencies. These roles are inspired by factuality and hallucination taxonomies [1, 2] and generate more diverse errors than simple entity or number substitutions.

For each role $k \in \mathcal{K}$, the generator first proposes a factual-error candidate and an explanation, as $(c^k_{t,0}, e^k_{t,0}) = G^k_{\theta^k_{t,0}}(x^+, E)$, where $\theta^k_{t,0}$ denotes the role-specific generation prompt at outer iteration t. Then another role is sampled as a reviewer, $a(k) \sim \text{Uniform}(\mathcal{K} \setminus \{k\})$, and provides a critique r^k_t, as $r^k_t = R^{a(k)}_\rho(x^+, E, c^k_{t,0}, e^k_{t,0})$. When the reviewer reports a vulnerability, SAFE-FEC updates the

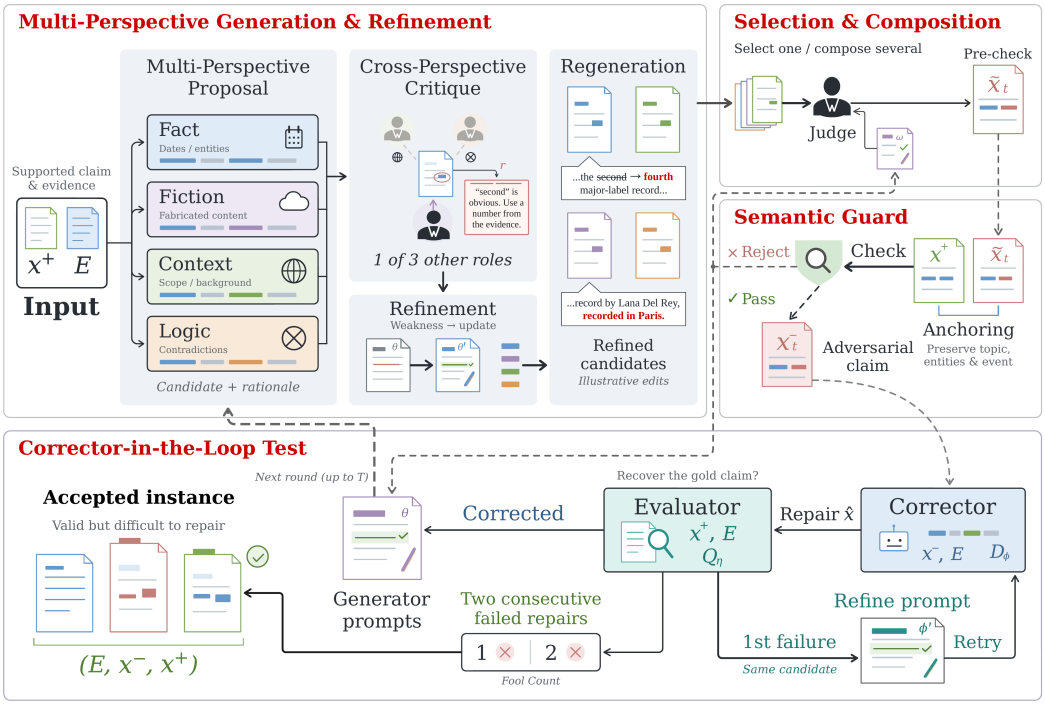

Figure 2: Overview of SAFE-FEC. Generators propose mutations from four perspectives; cross-perspective critiques guide conditional prompt refinement and candidate regeneration; a judge selects or composes one candidate; a semantic guard checks source anchoring; and an adaptive corrector tests designated-target recovery. Arrows indicate candidate transfer and feedback. Passing these operational filters does not independently establish that the mutation contradicts the evidence or that the target is fully supported.

role's generation prompt; otherwise it carries the prompt forward:

$$\theta^k_{t,1} = \begin{cases} U_G(\theta^k_{t,0}, r^k_t), & \text{if the reviewer identifies a weakness,} \\ \theta^k_{t,0}, & \text{otherwise.} \end{cases} \tag{1}$$

Subsequently, the generator produces a refined candidate as $(c^k_{t,1}, e^k_{t,1}) = G^k_{\theta^k_{t,1}}(x^+, E)$. The initial and refined candidate pools are: $\mathcal{C}_{t,0} = \{c^k_{t,0} : k \in \mathcal{K}\}, \mathcal{C}_{t,1} = \{c^k_{t,1} : k \in \mathcal{K}\}$. The reviewer's feedback can identify an obvious error, an unsupported addition, or semantic drift. The generator then makes a second proposal using the revised prompt. This is one critique-and-regeneration pass per role within each outer iteration; prompts are carried forward if another outer iteration is needed. The procedure adapts natural-language critique and refinement [14, 15] to FEC data construction.

3.3 Selection & Composition and Semantic Guard

After refinement, a mutation judge J_{ω_t} selects one candidate or proposes a composition of several candidates. The judge considers plausibility, evidence-checkability, topic consistency, and expected correction difficulty. This is a prompt-based assessment rather than an independent evidence-verification gate. The judge forwards only one selected or composed claim per outer iteration to the subsequent checks.

Formally, the judge outputs o_t as $o_t = J_{\omega_t}(x^+, E, \mathcal{C}_{t,1}) = (d_t, \mathcal{I}_t, \pi_t)$,

$$\tilde{x}_t = \begin{cases} c^j_{t,1}, & d_t = \text{select},\ \mathcal{I}_t = \{j\}, \\ \text{Merge}_\psi(\{c^j_{t,1} : j \in \mathcal{I}_t\}, x^+, \pi_t), & d_t = \text{compose}. \end{cases} \tag{2}$$

Here $d_t \in \{\text{select}, \text{compose}\}$ is the judge decision, $\mathcal{I}_t$ is the selected index set, and π_t is a merge recipe. When $d_t = \text{compose}$, the composer Merge_ψ follows the recipe to produce the pre-check mutation $\tilde{x}_t$.

The judge is instructed to avoid unchanged, unverifiable, or unnatural candidates. When it chooses composition, its recipe guides the composer to integrate complementary mutations into one coherent claim. The composed claim is checked in the same way as a selected claim; composition does not bypass the semantic guard.

However, judge selection alone is not enough. A fluent sentence may still drift away from the original correction task. SAFE-FEC therefore applies a semantic guard s_t, as

$$s_t = S_{\text{guard}}(\tilde{x}_t, x^+) \in \{0, 1\}. \tag{3}$$

The guard checks whether the candidate keeps the main subject, topic, and event frame of x^+ without remaining unchanged. If $s_t = 0$, the candidate is rejected and the failure reason updates the generator and judge prompts for the next outer iteration. Passing this check means that the candidate remains semantically anchored; it does not verify that the changed content contradicts E. Only guard-passing candidates enter the corrector test.

3.4 Corrector-in-the-Loop Test

For a candidate that passes the semantic guard, SAFE-FEC sets $x_t^- = \tilde{x}_t$ and tests it against an evidence-based detector-corrector D_ϕ, whose prompt or parameters are denoted by ϕ. Let ϕ_m denote the detector-corrector prompt or state at correction round m, with $\phi_1 = \phi$. At each round, the corrector produces an error-detection flag $\hat{z}_m \in \{0, 1\}$ and a correction $\hat{x}_m$:

$$(\hat{z}_m, \hat{x}_m) = D_{\phi_m}(x_t^-, E). \tag{4}$$

A correction evaluator Q_η, with evaluator prompt or parameters η, then checks whether the output recovers the meaning and factual scope of the designated target x^+ and is supported by E:

$$q_m = Q_\eta(\hat{x}_m; x^+, x_t^-, E) \in \{0, 1\}, \tag{5}$$

where $q_m = 1$ indicates recovery of the designated target. This is a strict operational criterion: an evidence-supported repair that is not equivalent in scope to x^+ can receive $q_m = 0$. We retain $\hat{z}_m$ for analysis, but use correction quality as the selection criterion. If $q_m = 1$, the candidate is discarded and evaluator feedback $\ell_{\text{eval},t}$ updates the generator prompts for the next outer iteration. If $q_m = 0$, the candidate survives one correction attempt and the detector-corrector prompt is updated:

$$\phi_{m+1} = U_D(\phi_m, \ell_{D,m}), \tag{6}$$

where $\ell_{D,m}$ contains the failed mutated claim and evidence, but does not reveal x^+, the exact edit span, or the gold correction. The revised corrector retries the same candidate; the corrector prompt is reset to ϕ for a new candidate.

For a candidate x, define the Correction Resistance Score over at most M correction rounds as

$$\text{CRS}_M(x; x^+, E) = \sum\nolimits_{m=1}^{M} \prod\nolimits_{r=1}^{m} (1 - q_r), \tag{7}$$

where $(\hat{z}_r, \hat{x}_r) = D_{\phi_r}(x, E)$ and $q_r = Q_\eta(\hat{x}_r; x^+, x, E)$. This score counts consecutive failures before the first target recovery; terms after a successful correction are zero and need not be evaluated. SAFE-FEC uses it as an *acceptance threshold*, not as an objective maximized over all generated candidates. It accepts the first guard-passing candidate with $\text{CRS}_M(x; x^+, E) \geq \kappa$, and otherwise continues to the next outer iteration, up to T iterations. In our implementation, $T = M = \kappa = 2$: the forwarded candidate must resist two consecutive attempts by the probing corrector. This establishes difficulty under that particular probe and strict target-recovery criterion, while the validity of accepted examples is assessed separately in Section 4 and Appendix I.

4 Experiments

We evaluate SAFE-FEC from four perspectives. First, we test whether SAFE-FEC examples reduce the correction performance of existing FEC systems. Second, we examine whether the generated examples remain valid and semantically grounded. Third, we ablate major components of SAFE-FEC. Finally, we provide a case study illustrating why SAFE-FEC produces more diagnostic examples than static corruption.

4.1 Experimental Settings

Benchmark Datasets. We evaluate model performance on three benchmark datasets: FECDATA, HOVER, and FEVEROUS. FECDATA [4] provides human-written FEC examples, while HOVER and FEVEROUS provide supported claims with multi-hop and heterogeneous evidence, respectively [10, 11]. FECDATA compares both Human vs. SAFE-FEC and CECoR vs. SAFE-FEC, whereas HOVER and FEVEROUS compare CECoR vs. SAFE-FEC [6]. Dataset construction details, statistics, and analysis are provided in **Appendix** D and **Appendix** F.

Correctors. We evaluate both general LLM correctors and FEC-specific correctors, including LIFE [5] and CECoR-style LLaMA correctors [6]. Table 1 uses abbreviated model names for readability; full model names are listed in **Appendix** E.

Evaluation Metrics. We report both LLM-based and automatic correction metrics. **Judge** is the average correction-success score assigned by three LLM evaluators. **J-GPT**, **J-DS**, and **J-Gem** denote the individual scores from GPT-4o, DeepSeek-V3, and Gemini-3-Flash evaluators. These scores measure whether the corrector recovers the gold claim, so lower values indicate worse correction performance. We also report ROUGE-2 and SARI. Since LLM-as-a-judge evaluation can exhibit biases, we report multiple judges and conduct pairwise quality evaluation [16, 17], complemented by a blinded human audit whose protocol and results are given in **Appendix** I.

Implementation Details. SAFE-FEC uses four generator personas: fact, fiction, context, and logic. The survival threshold is set to $\kappa = 2$, so a candidate is accepted only after surviving two consecutive detector-corrector attempts. The maximum number of outer generation-refinement loops is two. The specific prompts of each module are provided in **Appendix** L.

4.2 Main Results

Table 1 reports model-level correction performance before and after SAFE-FEC. SAFE-FEC substantially reduces correction performance across all three datasets. On FECDATA, where the original comparison set is human-written, the average Judge score drops by 36.9%, and the SARI score drops by 19.6%. This indicates that SAFE-FEC does not merely create artificial difficulty relative to synthetic baselines; it can also produce adversarial correction instances that are harder than human-written FEC examples for a broad range of correctors. The decrease is especially large for general LLM correctors, whose Judge scores fall from around 0.68–0.77 to 0.42–0.49, suggesting that SAFE-FEC exposes repair failures even in strong instruction-following models. Against CECoR-generated FECDATA, the average Judge drop over six correctors is even larger (53.4%), and all three judges agree (J-GPT -62.6%, J-DS -44.6%, J-Gem -54.1%).

The effect remains strong on HOVER and FEVEROUS, where the comparison baseline is CECoR-generated. Average Judge scores decrease by 36.0% on HOVER and 35.1% on FEVEROUS. These two datasets require multi-hop reasoning and heterogeneous evidence grounding, and the degradation shows that SAFE-FEC can generate errors that preserve surface plausibility while perturbing relations, temporal scope, contextual conditions, or evidence-composition links. Such examples remain close to the original claim, but repairing them requires more than local lexical substitution.

FEC-specific systems are also vulnerable. LIFE is already weak on FECDATA and HOVER, but it shows the largest degradation on FEVEROUS, dropping from 0.61 to 0.07 in Judge score and from 61.30 to 22.04 in SARI. This suggests that correction models trained on static synthetic supervision may fail sharply when multi-hop adversarial errors require structured-evidence grounding. LLaMA-SFT and LLaMA-GRPO also degrade under LLM-judge metrics. The main results show that SAFE-FEC consistently increases correction difficulty across human-written, synthetic, multi-hop, and structured-evidence settings. These drops do not come from reusing construction models: DeepSeek-V3, the only evaluated corrector and judge also used in construction, shows a smaller Judge drop than the six other correctors (-31.5% vs. -36.7%), and the two judges never used in construction report larger drops than the three-judge averages. Appendix G gives the full held-out analysis by corrector and by judge.

4.3 Dataset Quality Evaluation

A correction-performance drop alone does not show that SAFE-FEC examples are useful. Examples may be difficult because they are invalid, ambiguous, stylistically abnormal, or detached from the

Table 1: Main correction results. The FECDATA (vs. CECoR) block covers six correctors. Lower values indicate worse correction performance. Bold SAFE-FEC values indicate decreases relative to the corresponding original data. Average rows report relative change in percent.

Data	Type	Model	Orig.						SAFE-FEC					
			Jud.	J-GPT	J-DS	J-Gem	R2	SARI	Jud.	J-GPT	J-DS	J-Gem	R2	SARI
FECDATA (vs. Human)	LLM	Claude	0.77	0.74	0.78	0.80	0.79	82.29	**0.49**	**0.38**	**0.58**	**0.50**	**0.63**	**61.64**
		DeepSeek	0.70	0.65	0.73	0.72	0.70	76.12	**0.43**	**0.32**	**0.53**	**0.44**	**0.58**	**57.41**
		MiniMax	0.68	0.63	0.71	0.70	0.67	73.51	**0.42**	**0.32**	**0.52**	**0.43**	**0.54**	**55.06**
		Qwen3	0.71	0.66	0.73	0.74	0.70	74.77	**0.46**	**0.36**	**0.54**	**0.47**	**0.56**	**56.51**
	FEC	LIFE	0.07	0.07	0.07	0.07	0.09	27.10	**0.03**	**0.02**	**0.05**	**0.03**	**0.07**	**25.65**
		LLaMA-GRPO	0.58	0.53	0.62	0.59	0.55	62.02	**0.38**	**0.29**	**0.48**	**0.36**	**0.45**	**48.55**
		LLaMA-SFT	0.50	0.47	0.53	0.51	0.69	69.26	**0.41**	**0.34**	**0.48**	**0.40**	**0.64**	**61.73**
	Δ avg.		–	–	–	–	–	–	**-36.9**	**-48.4**	**-23.8**	**-38.5**	**-17.8**	**-19.6**
FECDATA (vs. CECoR)	LLM	DeepSeek	0.78	0.76	0.80	0.79	0.82	80.90	**0.43**	**0.32**	**0.53**	**0.44**	**0.58**	**57.41**
		MiniMax	0.78	0.75	0.81	0.78	0.80	79.58	**0.42**	**0.32**	**0.52**	**0.43**	**0.54**	**55.06**
		Qwen3	0.78	0.76	0.80	0.79	0.81	80.19	**0.46**	**0.36**	**0.54**	**0.47**	**0.56**	**56.51**
	FEC	LIFE	0.72	0.70	0.72	0.73	0.84	79.79	**0.03**	**0.02**	**0.05**	**0.03**	**0.07**	**25.65**
		LLaMA-GRPO	0.69	0.65	0.72	0.71	0.70	67.34	**0.38**	**0.29**	**0.48**	**0.36**	**0.45**	**48.55**
		LLaMA-SFT	0.78	0.76	0.79	0.80	0.88	84.80	**0.41**	**0.34**	**0.48**	**0.40**	**0.64**	**61.73**
	Δ avg.		–	–	–	–	–	–	**-53.4**	**-62.6**	**-44.6**	**-54.1**	**-41.2**	**-35.4**
HOVER	LLM	Claude	0.71	0.66	0.68	0.79	0.85	71.95	**0.51**	**0.41**	**0.56**	**0.56**	**0.79**	**57.69**
		DeepSeek	0.60	0.55	0.61	0.65	0.78	61.74	**0.42**	**0.33**	**0.53**	**0.41**	**0.70**	**47.27**
		MiniMax	0.62	0.56	0.64	0.67	0.71	53.94	**0.45**	**0.36**	**0.55**	**0.43**	**0.63**	**41.84**
		Qwen3	0.65	0.59	0.65	0.70	0.73	57.32	**0.47**	**0.38**	**0.56**	**0.47**	**0.64**	**43.58**
	FEC	LIFE	0.10	0.10	0.14	0.07	0.72	44.91	**0.05**	**0.05**	**0.09**	**0.02**	**0.21**	**18.95**
		LLaMA-GRPO	0.48	0.45	0.51	0.48	0.81	66.41	**0.29**	**0.24**	**0.39**	**0.25**	**0.67**	**50.01**
		LLaMA-SFT	0.49	0.46	0.49	0.53	0.84	72.60	**0.25**	**0.23**	**0.31**	**0.21**	**0.70**	**56.23**
	Δ avg.		–	–	–	–	–	–	**-36.0**	**-42.3**	**-22.1**	**-44.9**	**-20.8**	**-27.8**
FEVEROUS	LLM	Claude	0.84	0.82	0.83	0.88	0.88	75.57	**0.67**	**0.59**	**0.72**	**0.70**	**0.81**	**65.03**
		DeepSeek	0.81	0.78	0.81	0.83	0.83	69.90	**0.60**	**0.51**	**0.70**	**0.59**	**0.73**	**56.71**
		MiniMax	0.79	0.77	0.79	0.80	0.76	58.69	**0.60**	**0.51**	**0.70**	**0.58**	**0.66**	**48.69**
		Qwen3	0.74	0.72	0.75	0.75	0.71	56.02	**0.59**	**0.51**	**0.69**	**0.58**	**0.65**	**49.21**
	FEC	LIFE	0.61	0.60	0.62	0.60	0.81	61.30	**0.07**	**0.06**	**0.11**	**0.05**	**0.27**	**22.04**
		LLaMA-GRPO	0.62	0.64	0.70	0.53	0.73	58.87	**0.45**	**0.38**	**0.59**	**0.37**	**0.60**	**47.74**
		LLaMA-SFT	0.66	0.65	0.68	0.66	0.84	73.35	**0.40**	**0.35**	**0.52**	**0.33**	**0.72**	**58.37**
	Δ avg.		–	–	–	–	–	–	**-35.1**	**-43.2**	**-24.0**	**-38.8**	**-20.1**	**-23.6**

Table 2: Pairwise LLM-as-a-judge quality (%) evaluation. For FECDATA, the baseline is human-written FEC data; for HOVER and FEVEROUS, the baseline is CECoR.

Judge	FECDATA vs. Human			HOVER vs. CECoR			FEVEROUS vs. CECoR		
	SAFE	Tie	Base.	SAFE	Tie	Base.	SAFE	Tie	Base.
DeepSeek	**88.44**	0.15	11.40	**98.20**	1.65	0.14	**96.07**	2.85	1.07
Claude	**99.75**	0.10	0.15	**69.30**	1.05	29.65	**71.09**	0.87	28.03
Gemini	**71.11**	1.21	27.67	**66.93**	1.11	31.96	**66.81**	1.67	31.52
Macro avg.	**86.43**	0.49	13.07	**78.14**	1.27	20.58	**77.99**	1.80	20.21
Overall macro avg.	**80.86** / 1.18 / 17.95			SAFE-FEC / Tie / Baseline					

original correction target. We therefore conduct pairwise LLM-as-a-judge quality evaluation. For FECDATA, SAFE-FEC examples are compared against human-written FEC examples. For HOVER and FEVEROUS, SAFE-FEC examples are compared against CECoR-generated examples. Judges are asked to prefer the example that is more suitable as a challenging but valid FEC instance. Table 2 reports the preference results.

The pairwise evaluation provides a quality comparison, though an LLM preference cannot establish factual validity for every example. Across datasets and judges, SAFE-FEC is preferred in 80.86% of comparisons on average, while the corresponding baseline is preferred in 17.95%. Judges generally consider the generated examples suitable as challenging FEC instances, with low tie rates and a consistent direction across the listed settings. The human audit below examines factual status, evidence, and correction identifiability independently.

The preference is strongest on FECDATA, where SAFE-FEC is favored over human-written examples by 86.43% on average. This does not mean that human-written examples are low quality. Rather, it indicates that SAFE-FEC more directly targets diagnostic difficulty by evolving errors that current correctors fail to repair. On HOVER and FEVEROUS, SAFE-FEC is also preferred over CECoR by

Table 3: Ablation results on HOVER ($n = 100$ per configuration). Component names follow the SAFE-FEC framework in Section 3. Lower values indicate worse correction performance. Averages are over Claude, DeepSeek, and Qwen3; the lowest value in each column before rounding is bolded.

Variant	Claude						DeepSeek					
	Judge	J-GPT	J-DS	J-Gem	R2	SARI	Judge	J-GPT	J-DS	J-Gem	R2	SARI
w/o Multi-Perspective Generation	0.58	0.60	0.65	0.50	0.69	**60.31**	0.55	0.56	0.66	0.44	0.72	**53.47**
w/o Judge Selection	0.58	0.60	0.63	0.50	0.70	61.24	0.61	0.64	0.67	0.53	0.74	59.06
w/o Cross-Perspective Critique	**0.53**	**0.56**	**0.59**	**0.44**	**0.68**	60.58	0.55	**0.55**	0.64	0.47	**0.72**	56.28
w/o Adaptive Prompt Refinement	0.61	0.60	0.66	0.56	0.73	62.54	**0.53**	0.59	**0.60**	**0.42**	0.75	57.54

Variant	Qwen3						Average					
	Judge	J-GPT	J-DS	J-Gem	R2	SARI	Judge	J-GPT	J-DS	J-Gem	R2	SARI
w/o Multi-Perspective Generation	0.57	0.58	0.65	**0.48**	0.69	**49.02**	0.57	0.58	0.65	0.47	0.70	**54.27**
w/o Judge Selection	**0.57**	0.57	**0.61**	0.52	0.70	54.45	0.59	0.60	0.64	0.52	0.71	58.25
w/o Cross-Perspective Critique	0.57	**0.56**	0.67	0.49	**0.69**	53.54	**0.55**	**0.56**	**0.63**	**0.47**	**0.69**	56.80
w/o Adaptive Prompt Refinement	0.61	0.61	0.66	0.55	0.71	51.39	0.58	0.60	0.64	0.51	0.73	57.15

78.14% and 77.99%, respectively, suggesting that SAFE-FEC produces stronger adversarial FEC examples in multi-hop and heterogeneous-evidence settings. Because Table 2 is an LLM-as-a-judge evaluation, we also ran a blinded human audit: three graduate-student annotators independently labeled 300 SAFE-FEC examples (100 per dataset), seeing the designated gold claim only after locking their own minimal correction. By majority label, 98% of examples contain a factual error, 100% have sufficient evidence for a correction attempt, 99% have a unique correction target, and 100% remain semantically anchored. Fluency passes in 94%. Appendix I gives the protocol, per-annotator labels, and agreement.

4.4 Ablation Study

We ablate four components of SAFE-FEC on all three datasets: *Multi-Perspective Generation*, *Cross-Perspective Critique*, *Judge Selection and Composition*, and *Adaptive Prompt Refinement*. Table 3 reports the results on HOVER for Claude, DeepSeek, and Qwen3; the ablation results on FECDATA and FEVEROUS are given in **Appendix** J. Each ablated configuration independently constructs 100 examples, since changing the construction stages can alter which mutations pass the filters. These comparisons assess the resulting datasets, including differences in candidate generation and retention.

When correcting examples constructed without Adaptive Prompt Refinement, Claude and Qwen3 attain their highest Judge scores among the four ablations; this configuration also yields the highest ROUGE-2 averaged over all three correctors. This variant retains diverse generators and cross-perspective critiques, but does not use the critiques to update generation prompts. The pattern is consistent with the intended role of refinement: identifying a weakness is insufficient unless that feedback changes the subsequent mutation. However, DeepSeek shows the opposite response, attaining its lowest Judge score under this configuration. Feedback-driven regeneration therefore does not yield a uniform difficulty ordering across correctors; the resulting errors can expose different weaknesses in their correction behavior.

Judge Selection and Composition shows a similar interaction. Removing it yields the highest average Judge score (0.59), driven primarily by DeepSeek, while Qwen3 finds this configuration hardest. Selection thus appears to shape which candidate errors reach evaluation, with different consequences for different correctors. Meanwhile, removing *Cross-Perspective Critique* gives the lowest average Judge score (0.55), whereas single-perspective generation gives the lowest SARI. These outcomes do not imply that critique or diversity is dispensable: critique also addresses unsupported additions and semantic drift, and a lower correction score alone cannot establish better construction quality. Taken together, the ablations suggest that SAFE-FEC's stages should be assessed jointly for the difficulty and validity of retained examples, rather than treated as independent mechanisms that each monotonically increase difficulty.

4.5 Case Study

Table 4 illustrates how SAFE-FEC differs from static corruption on the same supported claim. CECoR creates a direct negation over a listed film, while SAFE-FEC preserves the original claim and introduces an error that requires connecting evidence across sentences. The CECoR-style mutation

Table 4: Case study comparing a CECoR-style mutation with a SAFE-FEC mutation. Red marks surface-obvious corruption; green marks the properties targeted by SAFE-FEC; orange marks evidence cues that must be connected.

Item	Text	Error signal	Diagnosis
Evidence	**E1:** *One Percent More Humid* stars **Juno Temple**. **E2:** Juno Temple appeared in films including *Black Mass*, ..., *The Three Musketeers*, and *The Dark Knight Rises*.	Two evidence facts must be jointly considered.	The evidence supports both the original film list and Juno Temple's role in *One Percent More Humid*.
Original	Juno Temple appeared in films including *Black Mass*, ..., *The Three Musketeers*, and *The Dark Knight Rises*.	Supported claim.	This is the gold correction target x^+.
CECoR	Juno Temple appeared in *Black Mass*. ... Juno Temple appeared in *The Three Musketeers*. Juno Temple **did not appear** in *The Dark Knight Rises*.	**Explicit negation of a listed fact.**	The mutation rewrites the list into repetitive sentences and flips an item that is explicitly present in the original claim and evidence. The error is valid but easy to localize.
SAFE-FEC	Juno Temple appeared in films including *Black Mass*, ..., *The Three Musketeers*, and *The Dark Knight Rises*, **but not in *One Percent More Humid***.	**Localized false exclusion requiring evidence composition.**	The original claim is preserved, while the added clause contradicts a separate evidence sentence. The minimal repair is identifiable: remove or reverse the false exclusion.

is valid but surface-obvious: it negates a film that is explicitly listed in both the claim and the evidence. The factual inconsistency can thus be localized by comparing the negated appearance with the corresponding supported statement. SAFE-FEC instead keeps the supported claim and its film list intact, and appends a false exclusion involving *One Percent More Humid*. The corrector must preserve the appearances already supported by the evidence while examining whether the added exclusion is also consistent with it. This requires distinguishing the unchanged correct content from the newly introduced contradiction and connecting the added clause to the relevant evidence sentence. The mutation therefore better matches the construction objective of SAFE-FEC: semantically anchored, evidence-grounded, minimally repairable, and resistant to simple surface matching.

5 Conclusion

This paper investigates FEC data construction from the perspective of correction difficulty and semantic validity. We argue that effective FEC data should not only contain valid factual errors but also challenge the correction capabilities of existing models. To this end, we propose SAFE-FEC, a semantically constrained adversarial framework that shifts FEC data construction from predefined error injection to correction-aware adversarial exploration. By integrating Multi-Perspective Generation & Refinement, Selection & Composition, Semantic Guard, and Corrector-in-the-Loop Test, SAFE-FEC progressively identifies challenging correction instances while preserving their original semantic context. Experiments on three benchmark datasets demonstrate that SAFE-FEC consistently generates more challenging instances for both general LLMs and FEC-specific correctors. On FEC-DATA, the average Judge and SARI scores decrease by 36.9% and 19.6%, respectively, compared with human-written examples. Average Judge scores also decrease by 36.0% on HOVER and 35.1% on FEVEROUS compared with CECoR-generated examples. The construction ablations reveal interactions between feedback, candidate selection, and corrector-specific difficulty, while quality evaluations and a human audit assess the generated examples' suitability for factual error correction. Future work will focus on strengthening evidence-grounded validity verification, accommodating alternative valid corrections, and improving construction efficiency.

Acknowledgments and Disclosure of Funding

This work was supported by the Science and Technology Major Project of the Science and Technology Program of the Xizang Autonomous Region (XZ202601ZD0010) and the National Natural Science Foundation of China (No. 62302333, U23B2053). The authors declare no competing interests.

References

[1] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[2] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online, 2020. Association for Computational Linguistics.

[3] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: A large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

[4] James Thorne and Andreas Vlachos. Evidence-based factual error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3298–3309, Online, 2021. Association for Computational Linguistics.

[5] Xingwei He, Qianru Zhang, A-Long Jin, Jun Ma, Yuan Yuan, and Siu Ming Yiu. Improving factual error correction by learning to inject factual errors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18197–18205, 2024.

[6] Lei Zhu, Xiaobao Wang, Jianbiao Yang, Chenyang Wang, Dongxiao He, Longbiao Wang, and Jianwu Dang. CECoR: Correction-oriented synthetic data construction for factual error correction. *arXiv preprint arXiv:2605.02277*, 2026.

[7] Jiangjie Chen, Rui Xu, Wenxuan Zeng, Changzhi Sun, Lei Li, and Yanghua Xiao. Converge to the truth: Factual error correction via iterative constrained editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12616–12625, 2023.

[8] Farima Fatahi Bayat, Kun Qian, Benjamin Han, Yisi Sang, Anton Belyy, Samira Khorshidi, Fei Wu, Ihab F. Ilyas, and Yunyao Li. FLEEK: Factual error detection and correction with evidence retrieved from external knowledge. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 124–130, Singapore, 2023. Association for Computational Linguistics.

[9] Xingwei He, A-Long Jin, Jun Ma, Yuan Yuan, and Siu Ming Yiu. PivotFEC: Enhancing few-shot factual error correction with a pivot task approach using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9960–9976, Singapore, 2023. Association for Computational Linguistics.

[10] Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3441–3460, Online, 2020. Association for Computational Linguistics.

[11] Rami Aly, Zhijiang Guo, Michael Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. FEVEROUS: Fact extraction and verification over unstructured and structured information. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[13] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

[14] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594, 2023.

[15] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[16] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[17] Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[18] Chong Tian, Qirong Ho, and Xiuying Chen. A symbolic adversarial learning framework for evolving fake news generation and detection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 12296–12310, Suzhou, China, 2025. Association for Computational Linguistics.

Appendix

This appendix provides supporting material for the main paper. **Appendix** B states the construction conditions and summarizes notation in Table 5, and Appendix C gives the compact SAFE-FEC procedure in Algorithm 1. Appendix D reports evaluated split statistics and acceptance rates in Tables 6 and 7; Appendix E lists the model and shot setting of every module in Table 8 and maps model abbreviations in Table 9; Appendix F reports SAFE-FEC error distributions in Tables 10 and 11. Appendices G–K report the held-out analysis, the pairwise quality evaluation against CECoR on FECDATA, the human audit, additional controls, and uncertainty, decoding, and cost details. Appendix L summarizes all prompt templates in Tables 22 and 23, followed by the full construction and evaluation prompts.

A Limitations

SAFE-FEC has several limitations. First, the framework relies on LLM-based generators, judges, semantic guards, detector-correctors, and evaluators, so the generated examples may inherit the biases, blind spots, and instability of the underlying models. Multiple evaluators and a human audit help characterize quality but cannot validate every released example. Second, the semantic guard checks anchoring to the source claim without reading the evidence; there is no independent verification gate requiring the mutation to contradict the evidence or the source claim to be fully supported. Third, the internal evaluator and main scoring judges use the original claim as a strict correction target. An alternative evidence-supported repair can therefore be scored as a failure if it differs in meaning or scope from that target. Fourth, the three evaluated datasets do not cover specialized or rapidly changing domains. Finally, iterative construction costs more than one-shot corruption. Future work should add explicit evidence validation, evaluate acceptable alternative repairs, and reduce construction cost through smaller specialized modules or caching.

B Notation and Algorithm Details

Target properties and implemented checks. A well-posed instance should exhibit evidential contrast (the mutation is erroneous with respect to E), semantic anchoring (it preserves the source entity and event), and correction resistance (a capable corrector struggles to recover the intended target). The implementation does not independently verify all three properties. Evidence-conditioned generation and judge instructions encourage evidential contrast; the guard directly checks anchoring to x^+; and the corrector test measures resistance to recovery of x^+ under a strict evaluator. The guard runs before the corrector, so semantic-guard rejection is not counted as correction resistance. A candidate passes the implemented filters when it is anchored and the probing corrector fails twice consecutively. The human audit in Appendix I provides an independent sample-level assessment: by majority label, 98% of audited examples contain a factual error, 100% have sufficient evidence for a correction attempt, 99% have a unique target, and 100% remain anchored.

Notation. Table 5 summarizes the notation used throughout Section 3. We group symbols by functional role so that readers can quickly locate notation for instance construction, multi-perspective generation, cross-perspective critique, selection and composition, semantic guarding, and corrector-in-the-loop testing.

Table 5: Compact notation summary for SAFE-FEC.

Group	Symbols	Meaning
Instance and objective	E, x^+, x^-, (E, x^-, x^+), $\mathrm{CRS}_M(x; x^+, E)$	E is evidence; x^+ is the source claim designated as the correction target; x^- is the mutation; and (E, x^-, x^+) is the constructed FEC instance. CRS_M counts consecutive failures to recover x^+ and is used as an acceptance threshold, not maximized over candidates.
Roles and generation	$\mathcal{K}$, k, $a(k)$, G, $G^k_{\theta^k_{t,b}}$, $\theta^k_{t,0}$, $\theta^k_{t,1}$, $\theta^k_{t+1,0}$	$\mathcal{K} = \{\mathrm{fact}, \mathrm{fiction}, \mathrm{context}, \mathrm{logic}\}$ is the generator-role set; $k \in \mathcal{K}$ is a generator role; $a(k) \sim \mathrm{Uniform}(\mathcal{K} \setminus \{k\})$ is the reviewer role. $G^k_{\theta^k_{t,b}}$ is the role-specific generator with prompt/state $\theta^k_{t,b}$, where $b = 0$ and $b = 1$ denote before and after critique. $\theta^k_{t+1,0}$ is the prompt carried to the next iteration.
Candidates and pools	$c^k_{t,0}$, $e^k_{t,0}$, $c^k_{t,1}$, $e^k_{t,1}$, $\mathcal{C}_{t,0}$, $\mathcal{C}_{t,1}$, $\tilde{x}_t$, x_t^-	$c^k_{t,0}$ and $e^k_{t,0}$ are the initial candidate and explanation from role k; $c^k_{t,1}$ and $e^k_{t,1}$ are the regenerated candidate and explanation after prompt refinement. $\mathcal{C}_{t,0}$ and $\mathcal{C}_{t,1}$ are the initial and refined candidate pools. $\tilde{x}_t$ is the judge-selected or composed pre-check candidate, and x_t^- is the candidate entering corrector-in-the-loop testing.
Critique and refinement	R, $R^{a(k)}_\rho$, ρ, r^k_t, U_G, $\ell_{\mathrm{sem},t}$, $\ell_{\mathrm{eval},t}$	$R^{a(k)}_\rho$ is the cross-perspective reviewer with prompt/parameters ρ; r^k_t is its critique. U_G updates generator prompts using reviewer, semantic-failure, or evaluator feedback. $\ell_{\mathrm{sem},t}$ is feedback from semantic-guard failure; $\ell_{\mathrm{eval},t}$ is feedback when the corrector succeeds.
Judge and composition	J, J_{ω_t}, ω_t, U_J, o_t, d_t, $\mathcal{I}_t$, π_t, Merge, Merge_ψ, ψ	J_{ω_t} is the mutation judge with prompt/state ω_t; U_J updates the judge after semantic failure. The judge output is $o_t = (d_t, \mathcal{I}_t, \pi_t)$, where $d_t \in \{\mathrm{select}, \mathrm{compose}\}$, $\mathcal{I}_t$ is the selected index set, and π_t is the merge recipe. Merge_ψ composes selected candidates using prompt/parameters ψ.
Semantic anchoring	S_{guard}, s_t	$S_{\mathrm{guard}}(\tilde{x}_t, x^+) \in \{0, 1\}$ checks whether $\tilde{x}_t$ remains semantically anchored to x^+. $s_t = 0$ rejects the candidate and triggers semantic feedback; $s_t = 1$ allows corrector-in-the-loop testing.
Corrector loop	D, D_ϕ, D_{ϕ_m}, ϕ, ϕ_m, $\hat{z}$, $\hat{x}$, $\hat{z}_m$, $\hat{x}_m$	D_ϕ is the initial detector-corrector; D_{ϕ_m} is its round-m version, with $\phi_1 = \phi$. It outputs a detection flag and correction, generically $(\hat{z}, \hat{x})$, or at round m, $(\hat{z}_m, \hat{x}_m)$.
Evaluation and survival	Q, Q_η, η, q_m, q_r, U_D, $\ell_{D,m}$, T, M, κ	Q_η evaluates whether the correction $\hat{x}_m$ recovers the designated target x^+ under E. $q_m = 1$ means target recovery, and $q_m = 0$ means failure under this criterion. q_r is the round-r value used in CRS. U_D updates the detector prompt using detector-failure feedback $\ell_{D,m}$. T is the outer-loop limit, M is the correction-round limit, and κ is the survival threshold.

C SAFE-FEC Algorithm

Algorithm 1 provides the compact procedural version of SAFE-FEC. Compared with the conceptual description in Section 3, the algorithm makes explicit how generator prompts, judge states, semantic feedback, evaluator feedback, and detector feedback are propagated across iterations.

Algorithm 1 SAFE-FEC

Input: claim x^+, evidence E, roles $\mathcal{K}$, operators $G, R, J, \mathrm{Merge}, S_{\mathrm{guard}}, D, Q, U_G, U_J, U_D$, prompts $\{\theta^k_{1,0}\}_{k\in\mathcal{K}}, \rho, \omega_1, \psi, \phi, \eta$, limits T, M, threshold κ.
Output: accepted instance (E, x^-, x^+), or no accepted candidate.

for $t = 1, \ldots, T$ **do**
  Propose: for each $k \in \mathcal{K}$, generate $(c^k_{t,0}, e^k_{t,0}) \leftarrow G^k_{\theta^k_{t,0}}(x^+, E)$.
  $\mathcal{C}_{t,0} \leftarrow \{c^k_{t,0} : k \in \mathcal{K}\}$.
  Critique and refine: for each $k \in \mathcal{K}$, sample $a(k) \sim \mathrm{Uniform}(\mathcal{K} \setminus \{k\})$, obtain $r^k_t \leftarrow R^{a(k)}_\rho(x^+, E, c^k_{t,0}, e^k_{t,0})$, and set

$$\theta^k_{t,1} \leftarrow \begin{cases} U_G(\theta^k_{t,0}, r^k_t), & \text{if } r^k_t \text{ identifies a weakness,} \\ \theta^k_{t,0}, & \text{otherwise.} \end{cases}$$

  Regenerate: for each $k \in \mathcal{K}$, generate $(c^k_{t,1}, e^k_{t,1}) \leftarrow G^k_{\theta^k_{t,1}}(x^+, E)$; set $\mathcal{C}_{t,1} \leftarrow \{c^k_{t,1} : k \in \mathcal{K}\}$.
  Default carry-over: $\theta^k_{t+1,0} \leftarrow \theta^k_{t,1}$ for all k, and $\omega_{t+1} \leftarrow \omega_t$.
  Judge: $o_t = (d_t, \mathcal{I}_t, \pi_t) \leftarrow J_{\omega_t}(x^+, E, \mathcal{C}_{t,1})$.
  Set $\tilde{x}_t \leftarrow c^j_{t,1}$ if $d_t = \mathrm{select}$ and $\mathcal{I}_t = \{j\}$; otherwise set

$$\tilde{x}_t \leftarrow \mathrm{Merge}_\psi(\{c^j_{t,1} : j \in \mathcal{I}_t\}, x^+, \pi_t).$$

  Semantic guard: $s_t \leftarrow S_{\mathrm{guard}}(\tilde{x}_t, x^+)$.
  if $s_t = 0$ **then**
    Construct $\ell_{\mathrm{sem},t}$; update $\theta^k_{t+1,0} \leftarrow U_G(\theta^k_{t,1}, \ell_{\mathrm{sem},t})$ for all k, and $\omega_{t+1} \leftarrow U_J(\omega_t, \ell_{\mathrm{sem},t})$.
    continue
  end if
  Corrector test: set $x^-_t \leftarrow \tilde{x}_t$, $\phi_1 \leftarrow \phi$, and $\mathrm{survival} \leftarrow 0$.
  for $m = 1, \ldots, M$ **do**
    $(\hat{z}_m, \hat{x}_m) \leftarrow D_{\phi_m}(x^-_t, E)$.
    $q_m \leftarrow Q_\eta(\hat{x}_m; x^+, x^-_t, E)$.
    if $q_m = 1$ **then**
      Construct $\ell_{\mathrm{eval},t}$; update $\theta^k_{t+1,0} \leftarrow U_G(\theta^k_{t,1}, \ell_{\mathrm{eval},t})$ for all k.
      break
    else
      $\mathrm{survival} \leftarrow \mathrm{survival} + 1$.
      if $\mathrm{survival} \geq \kappa$ **then**
        return (E, x^-_t, x^+).
      end if
      Construct $\ell_{D,m}$; update $\phi_{m+1} \leftarrow U_D(\phi_m, \ell_{D,m})$.
    end if
  end for
end for
return no accepted candidate.

D Dataset and Construction Details

Table 6 reports the evaluated split statistics and the corresponding SAFE-FEC adversarial construction statistics. We separate the original evaluated split from the generated adversarial subset to make clear which supported claims are used as correction targets and how many adversarial FEC instances are produced. For HOVER and FEVEROUS, we construct SAFE-FEC examples from supported claims in the evaluated split. For FECDATA, which already provides FEC pairs, we use the original correct sentence of each example as x^+ and generate a new adversarial erroneous input x^- from it.

Table 7 reports the end-to-end acceptance rate, defined as the proportion of input claims that are retained after the complete pipeline has been run over the entire evaluation split. More than half of the input claims are discarded: they either fail the semantic guard or are repaired on the first corrector-in-the-loop attempt, and so never enter the final dataset. In our metered runs, the semantic guard rejected 8 of 122, 13 of 119, and 3 of 109 forwarded candidates on FECDATA, HOVER, and FEVEROUS, and across the 1,200 construction-ablation examples 33.4% of guard-passing candidates were repaired by the detector on the first attempt and discarded.

Table 6: Evaluated split statistics and SAFE-FEC adversarial construction statistics. **SUP** and **REF** denote `SUPPORTS` and `REFUTES`, respectively. For SAFE-FEC, **REF** corresponds to the generated adversarial label.

Dataset	Split	Hops	Original evaluated split			SAFE-FEC adversarial set		
			SUP	REF	Total	SUP	REF	Total
FECDATA	Test	–	1593	2289	3882	0	1973	1973
HOVER	Val	2	521	605	1126	0	271	271
		3	968	867	1835	0	405	405
		4	511	528	1039	0	248	248
		Total	2000	2000	4000	0	924	924
FEVEROUS	Test	–	1411	1551	2962	0	633	633

Table 7: End-to-end acceptance of the released SAFE-FEC data.

Data (evaluated split)	Input claims	Finally accepted	Acceptance
FECDATA (test; correct sentence of every example)	3882	1973	50.8%
HOVER (val; SUPPORTS claims)	2000	924	46.2%
FEVEROUS (test; SUPPORTS claims)	1411	633	44.9%
Total	7293	3530	48.4%

E Model and Evaluator Details

Table 8 lists the model used by every module in construction and evaluation, together with its shot setting. Every module is zero-shot: no module receives in-context demonstrations or a worked example. The initial prompt templates are shared across FECDATA, HOVER, and FEVEROUS. At runtime, module-specific placeholders are filled with claims, evidence, candidates, and feedback as appropriate; generator, judge, and detector prompts may then change through the refinement steps described in Section 3. The three fine-tuned FEC correctors run with their own training-time input format and receive no demonstrations.

Table 8: Model and shot setting of every SAFE-FEC module.

Module	Model	Shots
Construction		
Candidate generators	DeepSeek-V3	0
Cross-Perspective Reviewers	Claude Haiku 4.5	0
Generator / judge / detector prompt refiners	Claude Haiku 4.5	0
Mutation judge (select or merge)	DeepSeek-V3	0
Composer	DeepSeek-V3	0
Semantic guard	DeepSeek-V3	0
Corrector-in-the-loop detector	GPT-5-mini	0
Corrector-in-the-loop evaluator	GPT-5-mini	0
Evaluation		
LLM correctors	Claude, DeepSeek, MiniMax, Qwen3	0
FEC correctors	LIFE, LLaMA-GRPO, LLaMA-SFT	fine-tuned
Scoring judges	GPT-4o, DeepSeek-V3, Gemini-3-Flash	0

The main tables use abbreviated corrector and evaluator names for readability. Table 9 gives the corresponding full model or checkpoint names.

Table 9: Abbreviations and full names used in the main results.

Abbreviation	Full name
Correctors	
Claude	claude-opus-4.5
DeepSeek	deepseek-v3
MiniMax	minimax-m2.7
Qwen3	qwen3-max
LIFE	LIFE corrector with gold-evidence [5]
LLaMA-GRPO	CECoR-style LLaMA-1b corrector trained with SFT + GRPO [6]
LLaMA-SFT	CECoR-style LLaMA-1b corrector trained with SFT [6]
Evaluators	
J-GPT	GPT-4o
J-DS	DeepSeek-V3 evaluator
J-Gem	Gemini-3-Flash evaluator

F Error-Type Distribution

Tables 10 and 11 report the error-type distribution of SAFE-FEC-generated adversarial examples. We focus on primary labels, assigning each example to one dominant category and subcategory, so the reported counts are mutually exclusive and easy to compare across datasets.

Table 10: Primary major error-type distribution of SAFE-FEC adversarial examples. *Unclassified* examples received no valid category label.

Primary category	FECDATA	HOVER	FEVEROUS
Fact	1328 (67.3%)	734 (79.4%)	512 (80.9%)
Context	209 (10.6%)	74 (8.0%)	63 (10.0%)
Logic	79 (4.0%)	31 (3.4%)	14 (2.2%)
Fiction	357 (18.1%)	84 (9.1%)	43 (6.8%)
Unclassified	0 (0.0%)	1 (0.1%)	1 (0.2%)

Table 11: Primary subcategory distribution of SAFE-FEC adversarial examples. Each example is assigned one dominant subcategory; *Unspecified* examples have a primary category but no valid subcategory within it, and *Unclassified* examples received no valid category label.

Family	Primary subcategory	FECDATA	HOVER	FEVEROUS
Fact	Entity	315 (16.0%)	122 (13.2%)	76 (12.0%)
	Number	207 (10.5%)	49 (5.3%)	56 (8.8%)
	Relation	204 (10.3%)	240 (26.0%)	114 (18.0%)
	Time	602 (30.5%)	241 (26.1%)	210 (33.2%)
	Unspecified	0 (0.0%)	82 (8.9%)	56 (8.8%)
Context	Circumstance	57 (2.9%)	33 (3.6%)	31 (4.9%)
	Omission / Addition	152 (7.7%)	36 (3.9%)	29 (4.6%)
	Coreference	0 (0.0%)	2 (0.2%)	1 (0.2%)
	Other context	0 (0.0%)	3 (0.3%)	2 (0.3%)
Logic	Link / Comparison	50 (2.5%)	28 (3.0%)	11 (1.7%)
	Modality / Tense	5 (0.3%)	2 (0.2%)	0 (0.0%)
	Negation	24 (1.2%)	1 (0.1%)	3 (0.5%)
Fiction	Fabricated attribute / source	230 (11.7%)	61 (6.6%)	31 (4.9%)
	Fabricated event	48 (2.4%)	22 (2.4%)	12 (1.9%)
	Unspecified	79 (4.0%)	1 (0.1%)	0 (0.0%)
Unclassified	–	0 (0.0%)	1 (0.1%)	1 (0.2%)

G Held-Out Correctors and Judges

Every corrector in Table 1 is held out from the corrector used inside the construction loop (GPT-5-mini; Table 8). Six of the seven correctors are also unused in any construction stage, and DeepSeek-V3 is the single overlapping model. Table 12 groups the relative drops of Table 1 by this stricter criterion. Excluding DeepSeek does not weaken the effect: the six fully held-out correctors show a larger average Judge drop than the overlapping model itself (-36.7% against -31.5%), so the difficulty of our data does not come from reusing DeepSeek. Restricting the group further to the three held-out general-purpose LLMs (Claude, MiniMax, and Qwen3) still gives Judge drops of 36.6%, 27.8%, and 21.5% on FECDATA, HOVER, and FEVEROUS.

Table 12: Relative drop caused by SAFE-FEC, grouped by whether the corrector is used in any construction stage.

	Fully held-out (6 correctors)			Overlapping (DeepSeek)		
Data	Judge	ROUGE-2	SARI	Judge	ROUGE-2	SARI
FECDATA	-36.6%	-17.9%	-18.8%	-38.6%	-17.1%	-24.6%
HOVER	-37.0%	-22.6%	-28.5%	-30.0%	-10.3%	-23.4%
FEVEROUS	-36.7%	-21.4%	-24.4%	-25.9%	-12.0%	-18.9%
All three	**-36.7%**	**-20.6%**	**-23.9%**	**-31.5%**	**-13.1%**	**-22.3%**

The same question applies to the judges. GPT-4o and Gemini are used at no point during construction or selection, so both are fully held out, and DeepSeek-V3 is the single overlapping judge. Table 13 reads the drop per judge. On all three datasets the overlapping judge reports the smallest drop of the three, so its effect runs against us. Averaging only the two fully held-out judges gives drops of 43.5%, 43.6%, and 41.0% on FECDATA, HOVER, and FEVEROUS, larger than the three-judge averages in Table 1, so the conclusion does not rest on the shared judge.

Table 13: Relative Judge drop caused by SAFE-FEC, read per judge. GPT-4o and Gemini are used in no construction stage.

Judge	Held out	FECDATA	HOVER	FEVEROUS
J-GPT	yes	-48.4%	-42.3%	-43.2%
J-DS	no	-23.8%	-22.1%	-24.0%
J-Gem	yes	-38.5%	-44.9%	-38.8%
Two held-out judges only		*-43.5%*	*-43.6%*	*-41.0%*

H Pairwise Quality Evaluation against CECoR on FECDATA

Table 14: Pairwise LLM-as-a-judge quality evaluation under the Gemini judge, including the comparison against CECoR-generated FECDATA. Values are preference percentages.

Comparison	SAFE-FEC	Tie	Baseline
FECDATA vs. CECoR	**82.00**	2.00	16.00
FECDATA vs. Human	**71.11**	1.21	27.67
HOVER vs. CECoR	**66.93**	1.11	31.96
FEVEROUS vs. CECoR	**66.81**	1.67	31.52

Table 2 compares SAFE-FEC with human-written examples on FECDATA. We additionally compared SAFE-FEC with CECoR-generated FECDATA examples under the same pairwise protocol, using the Gemini judge only. Table 14 places this comparison next to the three comparisons of Table 2 under the same judge. Gemini prefers SAFE-FEC over CECoR-generated FECDATA in 82.00% of the pairs, above its preference for SAFE-FEC over the human-written FECDATA examples (71.11%) and

over CECoR on HOVER (66.93%) and FEVEROUS (66.81%), so this comparison is the strongest of the four under this judge.

I Human Audit

We invited three human annotators, all graduate students, to audit 300 SAFE-FEC examples (100 unique source claims per dataset) in a blinded two-stage protocol. Stage A shows only the mutation and the evidence, to establish whether the example is a well-posed correction problem: each annotator rates factual status, evidence sufficiency, correction identifiability, and fluency (1–5), and writes down the minimal correction they would make. Once those answers are locked, Stage B reveals the gold claim and asks whether the mutation still concerns the same entity and event (semantic anchoring), whether the gold claim is itself supported by the evidence, and whether the annotator's own Stage A correction matches the gold meaning. All three annotators labeled all 300 examples independently, in different orders, without seeing one another's answers. Each dimension is a multiclass label; we take the majority label and report in Table 15 the pass rate for the category named in each column header.

Table 15: Audit pass rates (%) per dataset and annotator. *Majority* uses the majority label of the three annotators.

Data	Evaluator	Factual error (contra./unsup.)	Evidence (full)	Identifiable (unique)	Anchored (yes)	Fluent (≥ 4)
FECDATA	Annotator A	100	100	98	100	100
	Annotator B	98	92	98	100	99
	Annotator C	99	100	98	99	100
	Majority	*99*	*100*	*99*	*100*	*100*
HOVER	Annotator A	98	100	98	99	72
	Annotator B	98	85	96	98	87
	Annotator C	99	100	99	100	94
	Majority	*99*	*100*	*100*	*100*	*87*
FEVEROUS	Annotator A	96	100	95	99	76
	Annotator B	94	95	94	100	94
	Annotator C	99	99	99	100	100
	Majority	*96*	*99*	*98*	*100*	*95*
All 300	Annotator A	98	100	97	99	83
	Annotator B	97	91	96	99	93
	Annotator C	99	100	99	100	98
	Majority	**98**	**100**	**99**	**100**	**94**

Across all 300 audited examples, the majority finds a factual error in 98%, sufficient evidence for a correction attempt in 100%, a unique correction target in 99%, and preserved semantic anchoring in 100%. Fluency passes in 94%. Across these five criteria, 277 examples (92.3%) satisfy all five, 295 (98.3%) satisfy at least four, and all 300 satisfy at least three. Stage B also recorded whether each annotator's independent correction matched the gold meaning; we do not report that outcome here, so the audit does not quantify agreement with the designated target.

Pass rates alone do not show whether the annotators agreed with each other, so Table 16 reports agreement per dimension. Four of the five dimensions are close to unanimous, and in that regime chance-corrected coefficients such as Fleiss' kappa become unstable, because the chance baseline approaches the observed agreement itself. We therefore report Gwet's AC1, which is designed for skewed label distributions, together with the mean pairwise agreement it is computed from. Agreement is highest for semantic anchoring, correction identifiability, and evidence sufficiency (AC1 0.94–0.99), followed by factual status (0.83). It is lowest for fluency (0.64), a graded judgment on which per-annotator pass rates also differ most.

Table 16: Inter-annotator agreement per audit dimension.

Dimension	Mean pairwise agreement	Gwet's AC1
Semantic anchoring	98.9%	0.989
Correction identifiability	95.6%	0.954
Evidence sufficiency	93.7%	0.935
Factual status	85.0%	0.827
Fluency	67.2%	0.636

J Additional Controls

Without corrector-in-the-loop selection. We rebuilt 100 examples per dataset with the identical pipeline but with corrector-in-the-loop (CIL) selection switched off, and evaluated them with the same correctors and judges. Table 17 reports correction performance on these independently constructed datasets. This configuration retains candidate generation, refinement, and semantic checks, but accepts candidates without testing whether the probing corrector can recover the designated target. It therefore separates the construction of plausible, semantically anchored errors from the explicit screening of their correctability. The HOVER results also show why correction difficulty should not be judged by lexical overlap alone: Qwen3 attains a higher Judge score than DeepSeek (0.493 versus 0.405), despite a lower SARI (40.03 versus 45.26). Lexical recovery and judged correction quality can thus rank correctors differently on the same constructed dataset. These results characterize the examples admitted without the correction-based acceptance test; the retained-versus-discarded comparison in Table 19 further examines whether that test identifies difficulty that transfers to other correctors.

Table 17: Correction performance on separately constructed data without corrector-in-the-loop (CIL) selection ($n = 100$ per dataset). Higher values indicate better recovery of the designated gold target. Averages are over the three correctors; *All three* averages over datasets.

Data	Corrector	Judge	ROUGE-2	SARI
FECDATA	Claude	0.553	0.640	62.31
	DeepSeek	0.508	0.592	58.72
	Qwen3	0.551	0.576	58.92
	Average	*0.537*	*0.602*	*59.98*
HOVER	Claude	0.537	0.782	55.93
	DeepSeek	0.405	0.689	45.26
	Qwen3	0.493	0.598	40.03
	Average	*0.478*	*0.689*	*47.08*
FEVEROUS	Claude	0.737	0.778	64.06
	DeepSeek	0.693	0.716	56.95
	Qwen3	0.666	0.643	50.61
	Average	*0.699*	*0.712*	*57.21*
All three	Claude	0.609	0.733	60.77
	DeepSeek	0.535	0.665	53.64
	Qwen3	0.570	0.605	49.85
	Average	**0.571**	**0.668**	**54.75**

Construction ablations on FECDATA and FEVEROUS. Table 18 compares the four construction configurations on FECDATA and FEVEROUS, with 100 independently constructed examples per configuration and averages over Claude and DeepSeek. On both datasets, removing *Adaptive Prompt Refinement* yields the highest average Judge and SARI scores. On FEVEROUS, it also gives the highest Judge score for each corrector. This pattern is consistent with the role of refinement in turning critiques into revised mutations: retaining diverse generators and explicit feedback alone does not ensure difficult examples. The wider Judge range on FEVEROUS (0.729–0.772) than FECDATA (0.594–0.613) suggests that the observed differences depend on the source setting.

Selection behaves differently across datasets. Without *Judge Selection and Composition*, FECDATA has the lowest average Judge, ROUGE-2, and SARI, whereas HOVER has the highest average Judge score. Candidate selection therefore cannot be understood as uniformly maximizing downstream correction difficulty; its outcome depends on the candidate pool and the criteria used to retain examples. Together with HOVER's differences across correctors, these results motivate assessing the stages as a coupled construction process. They do not establish that a component is unnecessary: correction scores alone omit validity and diversity, and independently constructed datasets do not isolate effects on matched instances.

Table 18: Construction ablations on FECDATA and FEVEROUS ($n = 100$ per configuration). Variant names follow Table 3; lower values indicate worse correction performance. Averages are over Claude and DeepSeek. Within each dataset and metric, the lowest value across the four configurations is bolded separately for each corrector and for the average.

		FECDATA			FEVEROUS		
Variant	Corrector	Judge	R2	SARI	Judge	R2	SARI
w/o Multi-Perspective Generation	Claude	**0.618**	0.747	72.39	**0.733**	**0.792**	68.92
	DeepSeek	0.580	0.694	68.40	0.725	0.808	**62.75**
	Average	*0.599*	*0.720*	*70.39*	***0.729***	*0.800*	***65.84***
w/o Judge Selection	Claude	0.627	**0.696**	**69.46**	0.744	0.795	**68.23**
	DeepSeek	**0.560**	**0.652**	**66.57**	**0.721**	**0.801**	63.88
	Average	***0.594***	***0.674***	***68.02***	*0.732*	***0.798***	*66.05*
w/o Cross-Perspective Critique	Claude	0.623	0.741	71.97	0.769	0.794	68.89
	DeepSeek	0.589	0.694	67.92	0.751	0.827	66.46
	Average	*0.606*	*0.718*	*69.94*	*0.760*	*0.811*	*67.68*
w/o Adaptive Prompt Refinement	Claude	0.628	0.750	73.49	0.787	0.799	69.40
	DeepSeek	0.597	0.683	68.36	0.756	0.823	67.71
	Average	*0.613*	*0.716*	*70.93*	*0.772*	*0.811*	*68.56*

Retained versus discarded candidates. To verify that the retained examples are not merely the ones that happen to be hard for the corrector used inside the loop, we scored the discarded and the retained candidates side by side with the three evaluation correctors, none of which is the construction-time detector (GPT-5-mini). Table 19 shows that the candidates discarded by CIL selection are 0.23 Judge points easier on both datasets, and the gap holds separately for each of the three correctors (-0.20 to -0.29). The selection therefore isolates genuinely harder examples: the difficulty it captures transfers to correctors that were never in the loop, rather than being specific to GPT-5-mini.

Table 19: Judge scores of candidates retained (survived the detector) and discarded (repaired by the detector) by corrector-in-the-loop selection, scored by three correctors that were never in the loop.

Data	Corrector	Retained	Discarded	Difference
HOVER ($n = 278/122$)	Claude	0.487	0.772	-0.285
	DeepSeek	0.502	0.704	-0.202
	Qwen3	0.515	0.728	-0.213
	Average	*0.501*	*0.734*	*-0.233*
FEVEROUS ($n = 234/166$)	Claude	0.665	0.889	-0.224
	DeepSeek	0.647	0.867	-0.220
	Qwen3	0.622	0.885	-0.263
	Average	*0.645*	*0.880*	*-0.236*

K Uncertainty, Decoding, and Cost

Uncertainty. Table 20 reports a paired bootstrap on FECDATA, resampling the source claims with 10,000 replicates and a fixed seed. The intervals are narrow and every one of them excludes zero, including for the corrector with the smallest effect.

Table 20: Paired bootstrap on FECDATA (1,826–1,973 claims per corrector; 10,000 replicates).

Quantity	Point estimate	95% CI
Δ avg. Judge over the 7 correctors	-36.9%	[-37.9%, -34.9%]
Δ avg. ROUGE-2	-17.8%	[-18.9%, -16.7%]
Δ avg. SARI	-19.6%	[-20.2%, -19.0%]
Judge, Claude	-37.2%	[-39.1%, -35.2%]
Judge, LLaMA-SFT (smallest effect)	-19.6%	[-23.8%, -15.4%]

Decoding, seeds, and model versions. Every LLM call in the pipeline uses each provider's default decoding configuration: we set no temperature, top-p, or top-k at any stage. A fixed seed (42) controls the Python-side randomness in our ablation and supplementary runs, including reviewer-role sampling, stratified sampling, blinded example ordering, and the bootstrap above. The construction-side models are pinned to fixed snapshots.

Cost. We measured per-call token usage at every stage and priced the whole study at current rates (Table 21). Constructing one accepted example takes about 28 LLM calls plus the corrector-in-the-loop rounds, roughly USD 0.064. Evaluation dominates the budget, since the main tables contain 118,064 correction outputs (62,170 of them from API correctors) and 354,192 judge scores. Overall, each released example costs about USD 0.31: roughly USD 0.06 to construct it and USD 0.25 to evaluate it across all correctors and judges.

Table 21: Estimated cost of the study at current API rates.

Stage	Scale	Estimated cost
Constructing the 3,530 released examples	$\sim$100k LLM calls	USD 226
Main correction and judging (Table 1)	62,170 corrections; 354,192 judgments	USD 722
Pairwise quality evaluation (Table 2)	10,590 judgments	USD 12
Construction ablations and their evaluation	1,200 examples	USD 108
Supplementary experiments added during review	17,209 metered calls	USD 37
Total	$\sim$472k correction/judging inferences	$\approx$ USD 1,100

L Prompt Templates

This section lists all prompt templates used in SAFE-FEC construction and evaluation. All prompts are used zero-shot and are identical across the three datasets; the model and shot setting of every module are listed in Appendix E. Table 22 summarizes where each prompt is used in the pipeline, and Table 23 lists the four generator personas. The complete templates are provided in the same order as the SAFE-FEC pipeline. Runtime placeholders such as `{original}`, `{evidence}`, `{mutated}`, and `{feedback}` are filled automatically.

L.1 Generator Personas

Table 23 lists the four personas used by role-specific generators in Multi-Perspective Generation.

L.2 Mutation Generation Prompt

The generator prompt below is instantiated once for each persona in Table 23. It asks the generator to produce a plausible, semantically anchored factual mutation and an explanation of the injected error.

Error-injected generator.

```
[Role: Error-injected Generator - {persona}]
```

Table 22: Prompt families used in SAFE-FEC.

Prompt family	Pipeline role	Output / usage
Generator	Multi-Perspective Generation	Produces a mutated claim, error type, and explanation.
Reviewer	Cross-perspective critique	Identifies weaknesses and provides fix suggestions.
Prompt refiners	Adaptive Prompt Refinement	Update generator, detector, or judge prompts from feedback.
Mutation judge	Judge selection	Selects one candidate or decides to compose multiple candidates.
Composer	Candidate composition	Produces a merged mutation following the judge recipe.
Semantic guard	Semantic anchoring	Rejects candidates that drift from the original claim or remain unchanged.
Detector-corrector	Corrector-in-the-loop testing	Attempts to detect and repair the mutation using evidence.
Internal evaluator	Resistance evaluation	Determines whether the detector-corrector recovered x^+.
LLM scoring judge	Main metric evaluation	Scores model corrections against the gold correction.
Quality judge	Dataset-quality evaluation	Compares SAFE-FEC mutations with baseline mutations.

Table 23: Generator personas used for Multi-Perspective Generation.

Role	Persona description
`fact`	Expert in Factual Errors (Dates, Names, Entities).
`fiction`	Expert in Hallucination (Fabricating events).
`context`	Expert in Contextual Errors (Add/Drop info).
`logic`	Expert in Logical Fallacies (Contradictions).

```
You are an expert in generating error-injected data, specializing in introducing factual errors into
↪  correct statements while aligning them with the provided evidence. Your task is to inject factual
↪  errors into the original statement strict depending on your persona in a way that makes the
↪  resulting incorrect statement difficult for detectors to identify as containing factual
↪  inaccuracies, thereby helping to construct a challenging dataset.

### Strict Rules
1. Errors should be noted on the basis of the original sentence. It is prohibited to be completely
↪  different from the original sentence.
2. The mutation MUST look plausible and low-detectability (hidden, non-obvious).
3. It is prohibited to generate mutated sentences that are too long, too short or exactly the same as
↪  the original sentence.
4. Make sure to note the corresponding questions according to your role.

### Your Task
- Based on the Evidence given to you below, inject errors into the Original sentence to generate
↪  mutated sentences
- Explain which error was injected and why it is hard to detect.

### Output (strict JSON only)
{
  "mutated": "...",
  "error_type": "{persona}",
  "explain": "..."
}

Original: {original}
Evidence: {evidence}
```

L.3 Cross-Perspective Reviewer Prompt

The reviewer prompt below implements the cross-perspective critique stage summarized in Table 22. It evaluates whether the mutation is valid, hidden, and strategically strong, and returns actionable feedback for generator refinement.

Cross-perspective reviewer prompt.

```
[Role: Reviewer - Specialist in {reviewer_role} Errors]

Your job is to attack the mutated sentence ONLY from the perspective of {reviewer_role}.

### What you must do
1. Evaluate whether the injected error is valid for this role (factual / contextual / logical /
↪  hallucination).
2. Determine whether the mutated sentence successfully hides the error.
3. Identify any vulnerability in the mutation strategy.
4. Provide a concrete, actionable fix suggestion for the Generator.

### Output (strict JSON only)
{
  "attack_successful": true/false,
  "vulnerability": "...",
  "fix_suggestion": "..."
}

Original: {original}
Evidence: {evidence}
Mutated: {mutated}
"error_type": "{persona}"
Generator_Intent: {explain}
```

L.4 Prompt Refinement Prompts

The refinement prompts below correspond to the adaptive refinement rows in Table 22. They update generator, detector, and judge prompts while preserving each component's original task and JSON schema.

Generator prompt refiner.

```
[Role: Generator Prompt Refiner]

You will revise the previous *mutation-generation* prompt strictly according to the reviewer feedback.
Your goal is to improve the error-injection generator so that it produces harder-to-detect,
↪  high-quality mutated sentences.

### Absolute Rules
- DO NOT change the core task of "injecting subtle, hard-to-detect errors."
- DO NOT introduce unrelated behaviors or remove required fields.
- DO NOT expand or shorten the prompt excessively.
- Only fix the vulnerability pointed out in the reviewer feedback.
- Preserve output format, JSON schema, persona behavior, and structural integrity.

### Output
Return a JSON object with the following structure:
{
  "refined_prompt": "<the NEW refined generator prompt text only>"
}

Reviewer Feedback: {feedback}
Old Prompt: {old_prompt}
```

Detector prompt refiner.

```
[Role: Detector Prompt Refiner]

You will revise the previous *error-detection* prompt strictly according to the reviewer feedback.
Your goal is to improve the detector so that it corrects factual errors more reliably and consistently.

### Absolute Rules
- DO NOT change the core task of "detecting and correcting factual errors using evidence."
- DO NOT alter the JSON schema or expected fields.
- DO NOT expand or shorten the prompt excessively.
- Only fix the specific vulnerability identified in the reviewer feedback.
- Preserve logic, constraints, and step-by-step behavior required for detection accuracy.

### Output
Return a JSON object with the following structure:
{
  "refined_prompt": "<the NEW refined detector prompt text only>"
}

Reviewer Feedback: {feedback}
Old Prompt: {old_prompt}
```

Judge prompt refiner.

```
[Role: Judge Prompt Refiner]

You will revise the previous *candidate-selection / merge-judgement* prompt strictly according to the
↪  reviewer feedback.
Your goal is to improve the judge so that it selects or merges mutated candidates more accurately and
↪  consistently.

### Absolute Rules
- DO NOT change the core task of "evaluating candidates and selecting/merging based on quality."
- DO NOT remove or alter the selection/merge logic or JSON schema.
- DO NOT expand or shorten the prompt excessively.
- Only patch the vulnerability identified by the reviewer feedback.
- Preserve reasoning constraints, safety, JSON structure, and evaluation principles.

### Output
Return a JSON object with the following structure:
{
  "refined_prompt": "<the NEW refined judge prompt text only>"
}

Reviewer Feedback: {feedback}
Old Prompt: {old_prompt}
```

L.5 Mutation Judge and Composer Prompts

The judge and composer prompts implement the selection and composition stage summarized in Table 22. The judge either selects one candidate or asks the composer to merge multiple candidates according to a recipe.

Mutation judge prompt.

```
[Role: Mutation Judge]

Your job:
1. Evaluate multiple mutated candidates.
2. Decide whether to "select" the best single candidate, or "merge" two or more candidates into a
↪  stronger mutation.
3. Judge based on: subtlety, evidence-checkability, plausibility, topic consistency.
4. It is strictly prohibited to choose sentences that are exactly the same as the original or have no
↪  internal errors

### Output (strict JSON only)
{
  "action": "select" | "merge",
  "selected_indices": [int, ...],
  "merge_recipe": "...",
  "reasoning": "..."
}

Original: {original}
Evidence: {evidence}
Candidates: {candidates}
```

Mutation composer prompt.

```
[Role: Mutation Composer]

You will follow Judge's merge_recipe exactly to produce the final mutated sentence.

Mutated_list:{mutated}
Original: {original}
Recipe: {recipe}

### Output (strict JSON only)
{
  "composed_mutated": "..."
}
```

L.6 Semantic Similarity Guard Prompt

The semantic guard prompt implements the semantic anchoring check in Table 22. It rejects candidates that drift away from the original claim or are unchanged.

Semantic similarity guard prompt.

```
[Role: Semantic Similarity Guard]

### Task
Check whether the mutated sentence stays generally on the same topic, entities, and event frames as the
↪  original sentence. Minor additions, omissions, or rephrasings are allowed, as long as the main
↪  meaning or topic is not significantly changed.

### Criteria
- True if the main subject(s) and core event/theme are preserved.
- True if small contextual changes or minor extra details are present, but overall topic remains
↪  consistent.
- False only if the mutated sentence introduces a major topic shift, unrelated entities, or completely
↪  changes the event or mutated sentence is same with original sentence.

### Output (strict JSON only)
{
  "is_similar": true/false,
  "reason": "Explain briefly why the sentence is considered similar or not."
}

Original: "{original}"
Mutated: "{mutated}"
```

L.7 Corrector-in-the-Loop Prompts

The detector-corrector and internal evaluator prompts implement the corrector-in-the-loop resistance test in Table 22. The detector attempts to repair the mutation using evidence, while the evaluator checks whether the repair recovers the gold correction.

Evidence-based detector-corrector prompt.

```
[Role: Evidence-Based Fact Checker]

### Task
1. Detect whether the mutated sentence contradicts the evidence.
2. Correct the error using ONLY the evidence.
3. If no error is found, return found_error = false, correction = NULL

### Output (strict JSON only)
{
  "correction": "...",
  "found_error": true/false
}

Mutated: {mutated}
Evidence: {evidence}
```

Internal correction evaluator prompt.

```
[Role: Correction Evaluator]
You need to determine whether the corrected sentence has exactly the same meaning as the original
↪  sentence.

Use the following inputs:
- The original sentence
- Evidence supporting the original sentence
- The mutated (error-injected) sentence
- The corrected sentence

Your task is to check that the corrected sentence:
1. Fully preserves the meaning of the original sentence
2. Does not add or omit any information from the original sentence
3. Is supported by the provided evidence
4. "detector_success" = true if the corrected sentence meet all the above requirements else
↪  "detector_success" = false

### Output (strict JSON only)
{
  "detector_success": true/false,
  "reason": "..."
}

Original: {original}
Evidence: {evidence}
Mutated: {mutated}
Correction: {correction}
```

L.8 LLM-Based Evaluation Prompts

The following prompts are used outside the construction loop for experimental evaluation. As summarized in Table 22, one prompt scores correction outputs against gold corrections, and the other compares candidate mutations for dataset-quality evaluation.

LLM correction-scoring judge prompt.

```
You are a strict evaluator for factual error correction.

You will receive:
1. an erroneous sentence,
2. a piece of supporting evidence,
3. a corrected sentence produced by a model,
4. the gold-standard correct sentence.

Your task is to evaluate whether the corrected sentence correctly transforms the erroneous sentence
↪  into the meaning expressed by the gold-standard sentence.
The gold-standard sentence is the authoritative target.

-------------------------
CORE EVALUATION PRINCIPLES
-------------------------

(1) Priority of the Gold Standard
The corrected sentence must stay strictly aligned with the meaning and factual scope of the
↪  gold-standard correct sentence.

(2) Correction Quality
The corrected sentence should remove or fix the mistaken content found in the erroneous sentence.
It must NOT keep false or unsupported information from the erroneous sentence.

(3) Evidence as Secondary Support
Use evidence only to determine whether information contradicts the gold-standard.
Evidence may support removing errors, but must NOT be used to add details beyond the gold-standard
↪  sentence.

(4) Fluency
The corrected sentence should be coherent and grammatically sound.

-------------------------
SCORING INSTRUCTIONS
-------------------------

Provide a single numerical score between 0 and 1 (inclusive), rounded to three decimals.

Guidelines:
- 1.000 = Perfect: fully matches the meaning and factual scope of the gold-standard sentence, removes
↪  all errors, introduces no new details, and is fluent.
- 0.000 = Incorrect: fails to fix the errors, contradicts the gold standard, or is factually wrong.
- Deduct points for:
  - Missing key elements from the gold-standard sentence
  - Adding any factual content not present in the gold-standard sentence
  - Introducing unsupported details
  - Partial correction

-------------------------
INPUT
-------------------------

Erroneous sentence:
{original_mutated}

Evidence:
{evidence}

Corrected sentence:
{res_ori}

Gold-standard correct sentence:
{original}

-------------------------
OUTPUT REQUIREMENTS
-------------------------

Output ONLY one number between 0 and 1, rounded to three decimals.
Do NOT output words or explanations.
```

Pairwise dataset-quality judge prompt.

```
You are evaluating candidate errors for constructing a high-quality, high-difficulty challenging
↪  Factual Error Correction (FEC) dataset.
```

```
Given one correct claim C, evidence E, and two candidate incorrect mutations A and B, decide which
↪  candidate is BETTER as the incorrect side paired with C.

Evaluate A and B independently first, then compare them. Use the following criteria:
1) Factual incorrectness: the candidate should be genuinely wrong with respect to the evidence and the
↪  meaning of C.
2) Topic consistency: the candidate should stay on the same topic, entities, and event/frame as C.
3) Fluency and naturalness: the candidate should read like a plausible sentence, not broken or absurd
↪  text.
4) FEC usefulness: the candidate should be a meaningful correction target, not a trivial artifact or a
↪  mostly harmless rewrite.
5) Difficulty: among otherwise valid FEC candidates, prefer the one that is as difficult as possible to
↪  correct, as long as it still remains clearly incorrect, natural, and within the scope of the
↪  original claim and evidence.

Important instructions:
- Give substantial weight to difficulty: prefer the candidate that creates a harder factual correction
↪  example, provided it still stays on the same topic as the original claim, remains grounded in the
↪  evidence context, and is clearly wrong rather than confusing or off-topic.
- Prefer the candidate that is overall the better FEC example after balancing correctness, naturalness,
↪  and difficulty.
- If both are about equally good, return TIE.

Return ONLY one token: A, B, or TIE.

C (original_correct_claim): {correct_claim}
A (candidate mutation A): {mutation A}
B (candidate mutation B): {mutation B}
E (evidence): {evidence}
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction state the paper's scope as constructing challenging yet valid FEC data, and the claimed contributions are supported by the method in Section 3, the main results in Section 4, the ablation study, and the dataset-quality evaluation. The claims are limited to evidence-grounded FEC construction and evaluation on FECDATA, HOVER, and FEVEROUS.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations in Appendix A: the reliance on LLM-based generators, judges, semantic guards, detector-correctors, and evaluators; the evaluation on three datasets that do not cover every factual-error setting; the absence of a separate symbolic evidence-verification gate; and the higher cost of the iterative corrector-in-the-loop process.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present theoretical theorems or formal proof claims. The formulas in Section 3 define the SAFE-FEC search objective and Correction Resistance Score rather than proving theoretical guarantees.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides dataset splits and construction statistics in Appendix D, model and evaluator details in Appendix E, notation and algorithmic details in Appendix B and Appendix C, and the full prompt templates in Appendix L. These details specify the data construction pipeline, evaluation setup, and LLM-based judging procedure needed to reproduce or verify the main experimental claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The submitted document contains a detailed algorithm description, statistical information of the dataset, and prompt templates. We plan to make our source code and data available in the code repository after the paper is published.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings are described in Section 4, including datasets, correctors, metrics, and SAFE-FEC configuration. Appendix D reports the evaluated splits and generated adversarial set sizes, Appendix E lists full model names, and Appendix L provides the prompts used for construction and evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Appendix K reports paired-bootstrap 95% confidence intervals for the specified FECDATA comparisons, using 10,000 source-claim resamples with seed 42.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The construction pipeline, the four API correctors (Claude, DeepSeek, MiniMax, and Qwen3), and all judges are accessed through provider APIs, so we report API usage and cost rather than hardware. Appendix K gives the scale and the estimated API cost of every stage (construction, main correction and judging, pairwise quality evaluation, construction ablations, and supplementary experiments), about USD 1,100 in total.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The work uses public or previously established research datasets and focuses on evaluating and improving evidence-grounded factual correction. The method generates controlled factual-error examples for robustness evaluation, and the paper discusses the dual-use risk of adversarial error generation together with safeguards such as semantic anchoring, evidence conditioning, and evaluation against correction validity.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper discusses the positive purpose of SAFE-FEC, namely constructing harder FEC data to train stronger correction systems and to test current ones, which supports applications such as retrieval-augmented generation and fact-checking. It does not include a separate discussion of negative societal impacts, as its scope is the data construction method and its evaluation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper does not release a new generative model. For generated adversarial examples, SAFE-FEC includes semantic anchoring, evidence-conditioned construction, corrector-in-the-loop evaluation, and dataset-quality checks to filter invalid or uncontrolled generations. Any released examples will be tied to evidence and gold corrections rather than presented as standalone misinformation.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring

that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original datasets and methods used, including FECDATA, HOVER, FEVEROUS, LIFE and CECoR, and lists the model names in Appendix E. The dataset we use and the methods employed are all open-source and comply with the relevant usage terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces SAFE-FEC-generated adversarial FEC examples. Their construction process, dataset statistics, error-type distributions, prompts, and evaluation protocol are documented in Appendix D, Appendix F, Appendix C, and Appendix L. If released, the assets will be accompanied by this documentation and appropriate usage notes.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [No]

Justification: The only human annotation is the blinded audit in Appendix I, carried out by three invited graduate students rather than crowd workers. Appendix I describes the two-stage protocol, what the annotators judge in each stage, and the label counted as a pass for each audited dimension, but the paper does not reproduce the full annotation instructions or report compensation.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The human audit in Appendix I is an annotation task on publicly available claims and Wikipedia evidence. It collects no personal information about the annotators and involves no risk beyond reading and labeling text, so we consider IRB review not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are a core part of SAFE-FEC, including multi-perspective generation, cross-perspective critique, judge selection, semantic checking, corrector-in-the-loop testing, and LLM-based evaluation. The method section describes these components, Appendix E lists the LLMs used, and Appendix L provides the full prompt templates.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.