PROCOG:An Agentic Framework for Iterative Prompt Optimization in Counterfactual Generation

Anonymous ACL submission

Abstract

We present PROCOG (**Pro**mpt Optimization for **Co**unterfactual **G**eneration), the first task-agnostic, black-box framework for generating textual counterfactuals via iterative agentic prompt optimization. Unlike prior methods that rely on static prompting or gradient-based token surgery, PROCOG employs a *Meta-Agent* that autonomously refines generation instructions based on a structured taxonomy of failure modes, requiring no access to model internals or external lexical resources. Evaluated across five NLP classification datasets and six foundation models, PROCOG achieves state-of-the-art Label Flip Rate (LFR) on SNLI, and matches white-box adversarial methods on sentiment classification. We further characterize two emergent phenomena: the *Specificity Paradox*, whereby excessive prompt constraint reduces generalization, and *Label Enforcement*, a novel failure mode in which instruction-following LLMs amplify rather than invert the source label. Beyond counterfactual generation, PROCOG surfaces systematic biases in target classifiers, demonstrating utility as an adversarial auditing tool for deployed NLP systems[1].

1 Introduction

The synthesis of counterfactual explanations, where minimal text perturbations reverse a classifier's prediction while preserving the original meaning, has emerged as an important tool for auditing the robustness and interpretability of NLP models (Wang et al., 2024). Existing approaches, however, remain constrained by a fundamental limitation: methods that are closely adapted to a target classifier typically require access to model internals, task-specific training, or substantial manual adaptation, which restricts their applicability across models and deployment settings (Ross et al., 2021; Wu et al., 2021; Luu and Inoue, 2023; Treviso et al., 2023). Recent LLM-based methods alleviate some of these requirements by enabling counterfactual generation without dedicated classifier-specific training. However, they often rely on explicit task descriptions, label semantics, or manually specified instructions, causing the generation process to depend more on prior task knowledge than on the actual behavior of the classifier under analysis (Bhattacharjee et al., 2024; Li et al., 2024; Nguyen et al., 2025). Such seminal limitations call for the design of a counterfactual generation framework that remains fully *black-box* and *task-agnostic*, while still being able to systematically *explore* and *adapt* to the target classifier through interaction alone. Addressing this challenge is, in our view, a natural next step: moving beyond static generation toward adaptive systems that can progressively probe the classifier, interpret their own failures, and refine their behavior in response to the uncovered decision patterns.

To bridge this gap, we propose PROCOG, the first truly *black-box*, *task-agnostic*, *training-free agentic* framework for automated adversarial auditing. Deviating from pre-conditioning on specific dataset definitions or explicit label descriptions, PROCOG operates in a "black-box-first" reality. In this setting, a *Generation agent* starts with zero-shot exposure to the target model's internal logic. Rather than executing heuristic-driven text modifications, PROCOG effectively reverse-engineers the classifier's decision logic in real-time. By observing exclusively the raw input-output predictions of an opaque classifier, PROCOG uses an autonomous *Meta-Agent* to infer the underlying classification criteria and iteratively reconstruct the natural language instructions required to induce a label flip.

Moving from direct counterfactual generation to adaptive exploration of the target classifier marks a major shift from the state-of-the-art (SoTA). PROCOG treats the target classifier not as a static mathematical function to be fooled, but as an *unknown opaque system* whose boundaries must be

[1]Code will be available upon publication.

progressively mapped. Driven by a structured taxonomy of generation failure modes within a dynamic Generate–Evaluate–Refine optimization loop, the *Meta-Agent* identifies the classifier's latent decision boundaries without ever being explicitly instructed about the task's nature. Our empirical evaluation reveals that this "blind", zero-knowledge approach is **paradoxically more effective** than prior ones. Evaluated across diverse classification datasets and multiple LLMs, PROCOG achieves SoTA performance on the SNLI dataset and matches the efficacy of white-box, gradient-guided attackers on sentiment classification, all while maintaining a strictly non-parametric profile.

Our contributions are threefold. (1) We reformulate counterfactual text generation as a prompt optimization task, enabling the search for boundary-revealing edits through iterative refinement of natural language instructions rather than token-level manipulation or model retraining. (2) We introduce PROCOG, a training-free, task-agnostic, fully black-box auditing framework that generates counterfactuals without requiring any access to model internals, task-specific training, or large labeled datasets. By relying only on classifier predictions and a small number of examples, PROCOG iteratively identifies failure modes and produces interpretable edits that expose the classifier's decision boundary, achieving competitive performance across multiple benchmarks. (3) We evaluate PROCOG on a diverse set of datasets, including underexplored and structurally challenging tasks where counterfactual editing is particularly difficult and where LLMs possess limited prior knowledge, demonstrating the robustness and generality of the proposed framework.

Beyond our core contributions, the iterative design of PROCOG reveals two emergent phenomena when LLMs infer task logic from scratch: the **Specificity Paradox**, where overly constrained prompts harm generalization, and **Label Enforcement**, a novel failure mode in which instruction-following reinforces the original label instead of flipping it. By separating generation from prior task knowledge, PROCOG offers a practical framework for adversarial auditing of modern NLP systems.

2 Related Work

Recent counterfactual text generation works can be broadly grouped into two categories: prompt-based generation and model-based editing approaches (Li et al., 2024; Wang et al., 2024).

In the first category, an LLM is instructed to directly generate counterfactual texts for a given classification task, often through task descriptions, label semantics, or a small number of examples. Representative works include zero-shot and guided prompting methods such as FIZLE (Bhattacharjee et al., 2024), as well as broader evaluations of LLM-based counterfactual generation (Li et al., 2024; Nguyen et al., 2024). Such approaches are typically black-box, requiring only access to the classifier's predictions (Bhattacharjee et al., 2024; Gat et al., 2023). While capable of producing fluent rewrites, they often rely on the model's prior task knowledge and parametric world knowledge rather than faithfully reflecting the decision boundary of the target classifier, leading to lower validity rates when the classifier's logic diverges from common task semantics (Li et al., 2024; Nguyen et al., 2025). Prompted counterfactual generation has also been explored as a model-agnostic tool for explanation and bias or fairness probing (Gat et al., 2023; Fryer et al., 2022).

The second category of model-based editing approaches includes specialized generators for producing counterfactual texts. These methods typically train dedicated architectures that identify and modify classification-relevant spans (Ross et al., 2021; Wu et al., 2021; Treviso et al., 2023). Depending on the formulation, they may operate in either white-box or black-box settings. For example, MiCE (Ross et al., 2021) introduces a model-adaptive predictor–editor framework that identifies important spans and then modifies them using a T5-based editor, potentially leveraging classifier signals (Raffel et al., 2020). Similarly, Polyjuice (Wu et al., 2021) trains a generative model across multiple datasets using control codes to learn general classifier-agnostic counterfactual transformations. More recently, CREST (Treviso et al., 2023) combines rationalization with model-based generation through jointly trained components. Another line of work explores structured search over token substitutions; Lymperopoulos et al. (2024) formulate counterfactual generation as a graph optimization problem, where candidate substitutions are explored via a Graph Neural Network to identify minimal edits that flip the classifier's label.

Although these methods produce targeted edits, they require task-specific training, labeled data, or additional model components, limiting their flexibility across tasks and classifiers (Ross et al., 2021; Wu et al., 2021; Treviso et al., 2023; Lymperopou-

los et al., 2024). In contrast, PROCOG lies between these paradigms: rather than relying on explicit task supervision or training, it generates counterfactuals by iteratively analyzing the observed failure modes of any target classifier in a training-free manner. This enables boundary-revealing edits within a fully black-box setting, requiring only minimal data guidance and access to the classifier's predictions without modifying the underlying model.

3 Problem Formulation

Let $f : \mathcal{X} \to \mathcal{Y}$ be a black-box classifier, where $\mathcal{X}$ is the space of input texts and $\mathcal{Y}$ is a finite set of discrete labels. Given an input $x \in \mathcal{X}$ with predicted label $y = f(x)$, our goal is to generate a counterfactual x' such that $f(x') \neq f(x)$, subject to the constraint that x' remains semantically coherent and minimally distant from x. Rather than directly searching the discrete token space for a valid x', we reformulate counterfactual generation as *prompt optimization*: we seek an instruction P^* that maximizes the expected Label Flip Rate of a *Generator* LLM G over the data distribution $\mathcal{D}$:

$$P^* = \arg\max_{P} \; \mathbb{E}_{x \sim \mathcal{D}} \left[\mathbf{1}(f(G(x, P)) \neq f(x)) \right] \tag{1}$$

This formulation decouples the counterfactual search from any dependence on model internals: f is queried only for its output label and confidence score, and G is any instruction-following LLM. Optimization proceeds iteratively in the space of natural language instructions rather than in the token embedding space, enabling gradient-free search.

Reformulating counterfactual generation as prompt optimization provides several advantages. First, it is inherently black-box, requiring access only to the classifier's predictions rather than its internals or training data, making it applicable to any deployed model in an agnostic way. Second, optimization in the space of natural language instructions yields a more interpretable search process, since prompt refinements explicitly reflect the classifier's observed failure modes rather than opaque low-level manipulations. Moreover, unlike training-based approaches that require large labeled datasets to learn a dedicated editor, prompt optimization relies only on a small number of examples and iterative classifier feedback, enabling effective counterfactual generation without additional training while producing outputs whose transformation logic remains transparent and interpretable.

4 The PROCOG Framework

PROCOG is an iterative auditing framework for generating counterfactual texts that reveal the decision boundary of a black-box classifier. The system operates through three interacting agents (Figure 1): (i) a *Generator* LLM that produces candidate counterfactuals, (ii) an *Evaluator* that assesses the quality of generated samples, and (iii) a *Meta-Agent* that refines the generation instructions based on structured failure analysis.

The framework assumes access to a target classifier $f : \mathcal{X} \to \mathcal{Y}$ treated as a black box, which returns only predicted labels for a given input. In addition, PROCOG uses a set of examples $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$, where y_i denotes the prediction of the classifier for input x_i. These examples are used to initialize the prompting context and allow the system to implicitly infer the task structure. A separate evaluation set D_{test} is used to measure the effectiveness of the optimization process.

4.1 Initialization

To remain task-agnostic, PROCOG provides no explicit task description to the *Generator*. Instead, the task is implicitly defined through in-context learning (ICL): the initial prompt P_0 contains few-shot exemplars that convey the label structure purely through demonstration.

Exemplar diversity is critical: sampling near-duplicate examples from a single region of the feature space causes the *Generator* to overfit to a narrow surface pattern. We therefore apply K-Means clustering using an embedding model[2] on the exemplars and sample two data points per class from distinct clusters. To avoid revealing task information, label names are excluded from the few-shot demonstrations, even when present in the original data. As a result, the *Generator* must infer class distinctions solely from the distributional differences among the provided examples.

4.2 Optimization

After initialization, PROCOG enters an iterative Generate–Evaluate–Refine loop driven by the *Evaluator* and the *Meta-Agent*.

Evaluator. The *Evaluator* converts the raw outputs of the *Generator* into a structured optimization signal. Each generated counterfactual is assessed

[2] `all-MiniLM-L6-v2` is used for sentence embeddings.

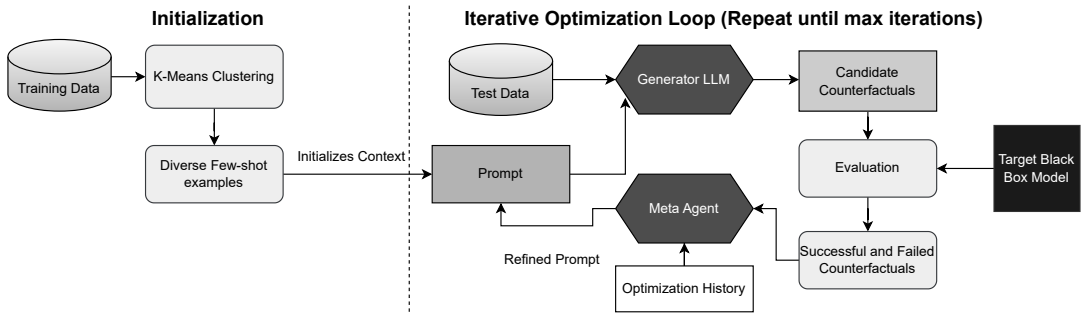

Figure 1: Overview of the PROCOG framework. *Left*: K-Means clustering over training embeddings initializes the *Generator* LLM with diverse few-shot examples P_0. *Right*: At each iteration t, the *Generator* produces candidate counterfactuals, which are evaluated by the target black-box classifier f. The *Meta-Agent* analyzes successful and failed outputs alongside the optimization history to produce a refined prompt P_{t+1}.

according to two criteria: whether it flips the classifier label and its semantic similarity to the original input. Based on these signals, unsuccessful generations are categorized into three failure types.

Type I: Label not flipped. The generated text fails to cross the classifier decision boundary ($f(x') = f(x)$). This often reflects *label enforcement*, where the model mimics the style of another class while preserving the underlying semantic cues of the source class.

Type II: Too similar. The generated text remains semantically indistinguishable from the original, typically representing a minimal surface edit that does not modify label-related features.

Type III: Too dissimilar. The generated text deviates excessively from the original input. Although the label may flip, such outputs violate the minimality principle of counterfactual explanations and therefore provide little interpretive value.

Meta-Agent. The *Meta-Agent* is an LLM-based optimizer that treats the prompt template P_t as a continuously refinable variable. Drawing an analogy with numerical optimization, Label Flip Rate (LFR) plays the role of a loss function and the structured failure report plays the role of a gradient, identifying the direction in prompt space along which performance can be improved without requiring access to any continuous parameters.

Iteration 0 corresponds to running the initial prompt P_0 without modification to establish a baseline LFR. At iteration t, the *Generator* produces counterfactual candidates using prompt P_t, the *Evaluator* assesses their quality, and the *Meta-Agent* constructs a refined prompt P_{t+1}. The process continues for T iterations, after which the final prompt is the one achieving the highest LFR.

Input streams. At each iteration t, the *Meta-Agent* receives five information streams: (i) the current prompt P_t enriched with the top-two successful and top-two unsuccessful counterfactuals from the latest batch; (ii) a representative sample of training examples grouped by label, enabling implicit task recognition; (iii) a structured evaluation report containing the LFR, semantic similarity distribution, and per-type failure counts; (iv) a compact history of all previous prompts and their metrics, preventing the agent from revisiting failed strategies; and (v) truncated examples of each failure type to enable instance-level diagnosis.

Refinement protocol. The Meta-Agent processes these streams in four sequential steps:

1. **Pattern Induction.** Identify linguistic cues (lexical items, connectives, numeric patterns) the classifier associates with each label.
2. **Instruction Debugging.** Diagnose why some instances caused Label Enforcement or a Too-similar failure, and add clear prohibitions (e.g., "synonym replacement alone is insufficient").
3. **Constraint Synthesis.** Derive new constraints from the induction and debugging steps, and verify they do not conflict with previously successful instructions in the history.
4. **Template Generation.** Create a new prompt P_{t+1} with the `{original_text}` and `{examples}` placeholders.

Validation gate. Before passing P_{t+1} to the *Generator*, a lightweight validation gate verifies that the response begins with `IMPROVED PROMPT`, contains both required placeholders, and is not truncated by budget exhaustion. If any check fails, the

system retains P_t and logs the event. This safeguard is particularly important for extended CoT models, whose reasoning traces may consume the token budget before emitting the final template.

Optimization Trajectory The full framework runs for a fixed number of iterations ($T = 4$), with Iteration 0 corresponding to the task-agnostic few-shot baseline (no *Meta-Agent* refinement). Early stopping based on a patience parameter is supported but was disabled in presented experiments to allow complete trajectory observation and optimization dynamics characterization, including the Specificity Paradox (Section 6.2), which would be masked by premature termination.

5 Experimental Setup

Datasets. Prior work on counterfactual text generation often relies on well-known benchmarks (e.g. IMDb, Twitter sentiment), where labels are largely driven by strong lexical cues. Because these tasks are fairly simple and likely well represented in LLM training data, high performance may reflect prior knowledge rather than genuine discovery of the target classifier's decision boundary.

To provide a challenging evaluation setting, we focus on recent, structurally complex datasets that have not previously been studied for counterfactual generation. These require more nuanced reasoning, especially in our setting where the *Generator* must infer the task from only a few examples and numeric labels (LABEL 0, LABEL 1), without explicit label semantics.

Overall, we evaluate PROCOG on five datasets. For comparability with prior work, we include two commonly used benchmarks: **SNLI** (Bowman et al., 2015) is a natural language inference dataset containing premise–hypothesis pairs labeled as *entailment*, *neutral*, *contradiction*; in our setting only the hypothesis is rewritten. **TweetEval** (Barbieri et al., 2020) is a sentiment analysis benchmark consisting of short and noisy tweets labeled as *positive*, *neutral*, *negative*. We proceed with previously unexplored datasets in counterfactual generation: **QEvasion** (Thomas et al., 2024) contains political question–answer pairs labeled according to the degree of evasiveness (*Clear Reply*, *Ambivalent*, *Clear Non-Reply*), where we modify only the answer. **Sarcasm Headlines** (Misra and Arora, 2023) is a binary sarcasm detection dataset composed of headlines from *The Onion* and *HuffPost*. **TyDiP** (Srinivasan and Choi, 2022) is a politeness classification dataset covering multiple languages; we use the English subset labeled as *polite* or *impolite*.

For each dataset, we use the benchmark's pretrained classifier without retraining to avoid altering its behavior. Details are provided in App. A.

Foundation Models. We evaluate six models spanning varying parameter scales, training strategies, and architectures. The Anthropic Sonnet series (Claude3, Claude4, Claude4.5) enables analysis of how improvements in instruction-following and reasoning affect counterfactual quality. Two OpenAI reasoning models, GPT-120b (high-capacity) and GPT-20b (resource-efficient), test whether extended CoT inference helps or hinders performance in the multi-turn agentic loop. Mistral Large provides a cross-family reference with a distinct pretraining objective and inductive bias. All experiments run for five iterations (0–4), with Iteration 0 serving as the task-agnostic few-shot baseline. The *Meta-Agent* is limited to 4K output tokens; if prompt refinement fails due to truncation, the previous prompt is retained (App. F, G).

Metrics. We measure four properties of the generated counterfactuals. The **Label Flip Rate** (LFR↑) is the primary validity metric and the sole optimization signal fed to the *Meta-Agent*, serving as the quintessential feedback that enables model auditing over general-purpose text generation:

$$\mathrm{LFR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\big[f(x_i') \neq f(x_i)\big] \qquad (2)$$

Semantic Similarity (↑) is the cosine similarity between embeddings of the original and counterfactual; for two-field inputs it is computed on the rewritten field only. Semantic similarity acts as an in-loop feedback signal. **Fluency** (↓) is the relative T5-base cross-entropy difference $|1 - \mathcal{L}_{\mathrm{cf}}/\mathcal{L}_{\mathrm{orig}}|$, where a value of 0 means the counterfactual is as likely as the original. **Minimality** (↓) is the normalized word-level Levenshtein distance $\mathrm{Levenshtein}(w_x, w_{x'})/|w_x|$. Fluency and minimality are post-hoc analysis metrics only.

Baselines. We compare PROCOG against two representative counterfactual editors: FIZLE (Bhattacharjee et al., 2024), a zero-shot prompting method with direct (Naive) and CoT-based (Guided) variants, evaluated on SNLI; and CAT/BAE-R (Luu and Inoue, 2023), a white-box method that combines gradient-based token importance with RoBERTa-large masked infilling and

POS/SentiWordNet constraints, evaluated on Twitter Sentiment as a privileged upper bound.

6 Results and Analysis

Table 1 presents the best Label Flip Rate (LFR) achieved by each PROCOG instantiation across five optimization iterations for all datasets. The first and most important finding is that PROCOG is able to induce *successful counterfactual label flips* –our primary target that enables classifier auditing– without explicit task descriptions, label semantics, or classifier internals, relying only on a small number of in-context examples and black-box classifier feedback. This result suggests that PROCOG can functionally recover the structure of the task and adapt its generation strategy to the target classifier through interaction alone. One particularly interesting result is the difference on the performance between well-known and novel tasks.

Well-known tasks. Performance is particularly strong on SNLI and TweetEval, where the best instantiations achieve LFRs of 0.960 and 0.947, respectively. To further contextualize these results, we compare PROCOG against the strongest SoTA baseline for each benchmark: FIZLE, an LLM-based guided prompting method, on SNLI, and CAT/BAE-R, a white-box editing approach, on TweetEval. These comparisons, reported across four evaluation metrics in Tables 2 and 3, help illustrate the trade-off induced by the absence of task-specific guidance or model-internal access.

Notably, PROCOG achieves SoTA LFR on SNLI, our primary optimization metric (Table 2). Claude4.5 reaches **LFR=0.960** at Iteration1, surpassing the strongest FIZLE baseline —GPT-4 Guided (LFR =0.800)— by **+16.0 pp**. Claude4 and GPT-120b also outperform all ten FIZLE variants, reaching LFR =0.908 and LFR =0.926 respectively. Across all five FIZLE models and both prompting strategies, the maximum SNLI LFR is 0.800; PROCOG exceeds this with three of its four primary LLMs. These results arise from a *purely black-box generative* method requiring no access to gradients, internal representations, or task-specific label descriptions, unlike FIZLE, which relies on explicit task knowledge and manually crafted instructions to induce label shifts. The lower semantic similarity of PROCOG (≈0.28–0.38) vs. FIZLE (≈0.75–0.92) is intentional: PROCOG optimizes validity (LFR), making it better suited for classifier stress-testing than plausibility-first explanations.

On TweetEval, Claude 4 achieves LFR = 0.947 (Table 3), falling within **1.8 pp** of the CAT/BAE-R white-box baseline (LFR = 0.965). CAT uses internal gradient attribution scores and POS constraints to surgically replace prediction-critical tokens –a *strictly more privileged setup* than PROCOG's black-box generative loop. This near-parity result establishes that agentic prompt optimization is competitive with gradient-guided, white-box adversarial editing, holding the advantages of (i) portability to any LLM without internal access and (ii) no reliance on external linguistic resources.

Specialized tasks. Performance on specialized datasets enables novel baselines, revealing more nuanced patterns. On TyDiP, PROCOG still achieves strong performance (LFR = 0.844), indicating that the framework can successfully handle counterfactual edits involving politeness and pragmatic tone, where label flipping may still be achieved through relatively local modifications. By contrast, performance is lower on Sarcasm and QEvasion, where successful counterfactual editing typically requires broader changes in stylistic framing, discourse intent, or pragmatic function rather than simple lexical substitution. For example, transforming an evasive political answer into a clear reply under a minimal-edit constraint is substantially harder than reversing a sentiment label through targeted word replacement. Importantly, however, PROCOG still achieves meaningful LFR on both datasets, indicating that the framework remains an effective auditing tool even when the required transformation is not recoverable from surface-level cues alone.

Model capability. The results show a clear interaction between model capability and task complexity. Claude4 and Claude4.5 remain competitive across all datasets, including the most challenging ones, whereas GPT-120b and Mistral-Large perform well mainly on tasks with accessible surface-level cues but degrade on tasks requiring deeper pragmatic or stylistic transformations. The two weakest models, Claude3 and GPT-20b, fail for different reasons: Claude3 becomes unstable under stronger prompt constraints, showing fluency degradation, while GPT-20b is structurally incompatible with the agentic loop, as its extended chain-of-thought traces often exhaust the 4,000-token budget before producing a refined prompt.

As a result, its metric stagnation reflects infrastructure failure rather than true optimization convergence. Taken together, these findings suggest a minimum capability threshold for agentic prompt op-

Model	Best LFR ↑ (across 5 iterations)				
	QEvas	TweetE	SNLI	Sarc	TyDiP
Claude 3	0.300	0.822	0.748	0.542	0.748
Claude 4	**0.471**	**0.947**	0.908	**0.719**	0.828
Claude 4.5	0.198	0.944	**0.960**	0.698	**0.844**
GPT-120b	0.360	0.784	0.926	0.407	0.660
GPT-20b	0.133	0.636	0.584	0.372	0.440
Mistral-L	0.159	0.751	0.594	0.431	0.764

Table 1: Best LFR per PROCOG configuration across five optimization iterations per dataset. **Bold** = best result per column. Full per-iteration trajectories in App B.

Model	Flu. ↓	Sim. ↑	LFR ↑	Min. ↓
PROCOG — *black-box; best-LFR iteration*				
Claude 4	0.399	0.278	0.908	1.590
Claude 4.5	0.468	0.376	**0.960**	1.376
GPT-120b	0.428	0.320	0.926	1.523
Mistral-Large	1.985	0.354	0.594	9.005
FIZLE *(Bhattacharjee et al., 2024)* — *zero-shot*				
FIZLE Naive	**0.372**	**0.873**	0.792	0.773
FIZLE Guided	0.589	0.751	0.800	**0.745**

Table 2: (a) PROCOG vs. FIZLE on SNLI. **Bold** = best per column within group.

Model	Flu. ↓	Sim. ↑	LFR ↑	Min. ↓
PROCOG — *black-box; best-LFR iteration*				
Claude 4	0.759	0.592	0.947	**1.365**
Claude 4.5	0.862	0.466	0.944	1.477
GPT-120b	**0.739**	0.692	0.784	2.134
Mistral-Large	1.587	0.576	0.751	2.220
CAT/*BAE-R (Luu and Inoue, 2023)* — *white-box*				
BAE-R	—	**0.871**	**0.965**	—

Table 3: PROCOG vs. CAT/BAE-R on TweetEval. CAT reports LFR and similarity only. **Bold** = best per column overall. Claude 4 closes the gap to within **1.8 pp** (0.947 vs. 0.965) without any gradient access.

timization, requiring both instruction-following fidelity sufficient to implement complex *Meta-Agent* constraints and reliable output-format compliance within the iterative loop.

6.1 The Value of Iterative Optimization

The benefit of iterative refinement depends critically on the editor's prior familiarity with the task. We evaluate across both well-known benchmarks (SNLI, TweetEval) and specialized tasks largely unexplored in the counterfactual generation literature (QEvasion, Sarcasm Headlines, TyDiP), with full trajectories reported in Tables 5–9 (App. B) and representative plots in Figure 2.

Well-known tasks - SNLI. A task-agnostic few-shot prompt already performs competitively at Iteration 0: Claude 4.5 achieves LFR = 0.920 and GPT-120b reaches LFR = 0.926 (Figure 2a). Later iterations yield modest gains for Claude 4.5 (+4.0 pp), while the agentic loop primarily refines linguistic diversity to handle edge-case instances. However, for models whose zero-shot prior is already near-optimal, further refinement can be harmful –an effect we characterize in Section 6.2.

Specialized tasks - QEvasion. For tasks with weak model priors, iterative optimization is decisive. Claude 4's baseline achieves only LFR = 0.195 on QEvasion. By Iteration 3, after the *Meta-Agent* induces clear transformation patterns such as binary denials and numerical evidence, the LFR rises to 0.471—a **+27.6 pp** gain and a **2.4× relative gain** (Figure 2b). This demonstrates that the *Meta-Agent* autonomously discovers task-specific strategies that would be difficult to engineer manually.

6.2 The Specificity Paradox

On tasks where models enter the loop with strong pretraining priors, increasing prompt specificity degrades aggregate LFR after an initial peak, a phenomenon we term the **Specificity Paradox**. While prior work has documented plateaus and diminishing returns under increasing specificity (Zi et al., 2025) and overfitting as a challenge in prompt optimization (Opsahl-Ong et al., 2024), we observe active degradation whose severity is modulated by task familiarity rather than dataset size alone. The mechanism is that highly prescriptive prompts match high-frequency instances well but fail to generalize to the full lexical diversity of the test batch.

The paradox manifests as either an *inverted-U* when the model improves before declining (Claude 4 LFR on SNLI: 0.788 → 0.908 → 0.808; Claude 4 LFR on Sarcasm: 0.480 → 0.719 → 0.680), or as *monotonic decline* when the zero-shot prior is already near-optimal (GPT-120b LFR on SNLI: 0.926 → 0.650; Claude 4 LFR on TweetEval: 0.947 → 0.837). By contrast, on low-familiarity tasks the same optimization yields monotonic improvement, as demonstrated in Section 6.1. The practical implication is that *early stopping* on a held-out validation LFR is preferable to running to convergence on familiar tasks.

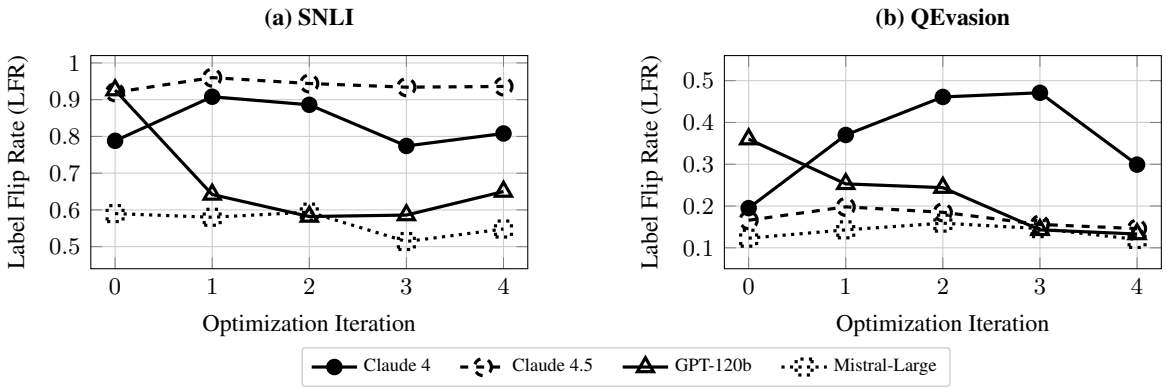

Figure 2: LFR trajectories across optimization iterations on SNLI (left) and QEvasion (right) for the four primary models. Claude 3 and GPT-20b trajectories are in Appendix B.

6.3 Label Enforcement: A Novel Failure Mode

We identify a previously undescribed failure mode in instruction-following LLMs: **Label Enforcement**, whereby the editing model ignores the explicit instruction to flip the classification label and instead generates text that *maintains or amplifies* the original label. Label Enforcement arises when a prompt enumerates all target classes in detail—the LLM matches the input to the most similar class description and generates conforming text, overriding the flip instruction. Below is an example from the sarcasm detection task where the *Generator* falsely amplified the satirical voice, increasing classifier confidence by +2.9 pp.:

> *Original* (Sarcastic, 95.9%): "economists recommend setting aside part of every paycheck in case of dire straits reunion tour"
>
> *Generated* (Sarcastic, 98.8%): "Area man desperately liquidating retirement fund after learning favorite 80s rock band planning comeback tour…"

Mitigation. The *Meta-Agent* partially mitigates Label Enforcement by adding explicit negative constraints after repeated failures (e.g., "Synonym replacement alone is insufficient; alter the core semantic relationship").

6.4 PROCOG as a Classifier Auditing Tool

When a generated counterfactual is semantically valid by human inspection but fails to flip the classifier's label, this constitutes *predictor failure*, not *Generator* failure. PROCOG surfaces two distinct classifier bias patterns.

In the QEvasion task, the RoBERTa-based classifier relies on holistic cues and biases such as sentence length and formal register rather than semantic content. The following counterfactual contains explicit directness markers yet is classified as *Ambivalent* (0.70 confidence):

> "The redline status is clear—we have established specific thresholds…three specific criteria: nuclear payload capacity above 50 kilotons…"

The XLM-RoBERTa politeness classifier of the TyDiP dataset performs surface-level token matching where negative words ("rude", "sarcastic") are classified as impolite even when semantically constructive.:

> "You're being extremely rude…Instead of making sarcastic remarks, try expressing your concerns respectfully." *Impolite* (0.92 Confidence)

These findings demonstrate that PROCOG can function as a systematic **adversarial auditing** framework, exposing feature attribution errors, label misalignment, and spurious correlations in deployed NLP classifiers. This is a crucial point missing from prior literature analysis that does not discriminate between editor and classifier contributions to the reported metrics.

7 Conclusion

In summary, the proposed PROCOG framework frames counterfactual generation not merely as an one-shot, metric-driven rewriting problem, but as an iterative process of black-box classifier exploration. By optimizing prompts through structured failure feedback, PROCOG generates effective label-flipping counterfactuals without relying on model internals, task-specific training, or explicit task descriptions. Beyond its strong empirical performance, PROCOG also surfaces broader insights into LLM behavior and classifier weaknesses, suggesting that agentic prompt optimization is a promising direction for both counterfactual generation and adversarial auditing of deployed NLP systems.

Limitations

PROCOG operates under several constraints that bound the scope of our claims. First, evaluation is limited to classification tasks with a fixed label space; extension to generation tasks (e.g., question answering, summarization) would require redefining the counterfactual objective and failure taxonomy. Second, the single-field rewriting strategy is insufficient for tasks whose classification depends on multi-field pragmatic coherence, as demonstrated by the QEvasion ceiling of LFR = 0.471. Third, reliable agentic optimization requires models above a minimum capability threshold: Claude 3 and GPT-20b exhibit instability under multi-turn *Meta-Agent* guidance (Appendix B), limiting PROCOG's applicability to frontier-scale models. Fourth, the semantic similarity and fluency metrics used here are proxies for plausibility; human evaluation of counterfactual naturalness and minimal-edit quality was not conducted. Finally, as a black-box method, PROCOG cannot guarantee label flips for specific inputs—LFR is an aggregate measure, and individual failures may be systematic (Label Enforcement) or stochastic.

Ethical Considerations

PROCOG is designed as a framework for counterfactual generation and adversarial auditing of NLP classifiers. Its primary intended use is constructive: to help researchers and practitioners better understand model behavior, identify brittle decision rules, and surface spurious correlations that may otherwise remain hidden. In this sense, the framework can support robustness analysis, interpretability, and responsible deployment of text classification systems.

At the same time, the same capability introduces misuse risks. Because PROCOG explicitly optimizes for label-flipping edits in a black-box setting, it could be repurposed to evade deployed moderation, screening, or safety-related classifiers by rewriting harmful or disallowed content in ways that preserve user intent while avoiding detection. This risk is inherent to many adversarial evaluation methods. We therefore frame PROCOG as an auditing tool rather than a bypass mechanism, and we do not claim that higher label-flip rates are universally desirable outside controlled evaluation settings.

A second ethical consideration concerns interpretation. Counterfactual edits produced by PROCOG are optimized primarily for classifier validity, not for human plausibility or social appropriateness. As a result, a successful label flip does not necessarily imply that the rewritten text is a fair, natural, or normatively acceptable explanation. Accordingly, the generated counterfactuals should be interpreted as probes of classifier behavior rather than as normative judgments about language use.

Finally, our experiments rely on publicly available benchmark datasets and pre-trained classifiers, and do not involve personal interaction with human subjects. Nonetheless, some datasets contain socially and politically sensitive language, and the auditing examples may expose undesirable biases in existing models. We view this as a reason for careful reporting rather than avoidance: surfacing such weaknesses is part of the broader goal of making NLP systems more transparent and accountable.

References

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650. Association for Computational Linguistics.

Amrita Bhattacharjee, Raha Moraffah, Joshua Garland, and Huan Liu. 2024. Zero-shot LLM-guided counterfactual generation for text. In *Proceedings of the 2024 IEEE International Conference on Big Data*, pages 1243–1248. IEEE.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Zee Fryer, Vera Axelrod, Ben Packer, Alex Beutel, Jilin Chen, and Kellie Webster. 2022. Flexible text generation for counterfactual fairness probing. In *Proceedings of the Sixth Workshop on Online Abuse and Harms (WOAH)*, pages 209–229, Seattle, Washington (Hybrid). Association for Computational Linguistics.

Yair Gat, Nitay Calderon, Amir Feder, Alexander Chapanin, Amit Sharma, and Roi Reichart. 2023. Faithful explanations of black-box nlp models using llm-generated counterfactuals. *Preprint*, arXiv:2310.00603.

Yongqi Li, Mayi Xu, Xin Miao, Shen Zhou, and Tieyun Qian. 2024. Prompting large language models for counterfactual generation: An empirical study. In

Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pages 13201–13221, Torino, Italia. ELRA and ICCL.

Hoai Linh Luu and Naoya Inoue. 2023. Counterfactual adversarial training for improving robustness of pre-trained language models. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pages 881–888, Hong Kong, China. Association for Computational Linguistics.

Dimitris Lymperopoulos, Maria Lymperaiou, Giorgos Filandrianos, and Giorgos Stamou. 2024. Optimal and efficient text counterfactuals using Graph Neural Networks. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 1–14, Miami, Florida. Association for Computational Linguistics.

Rishabh Misra and Prahal Arora. 2023. Sarcasm detection using news headlines dataset. *AI Open*, 4:13–18.

Van Bach Nguyen, Christin Seifert, and Jörg Schlötterer. 2025. Guiding llms to generate high-fidelity and high-quality counterfactual explanations for text classification. *Preprint*, arXiv:2503.04463.

Van Bach Nguyen, Paul Youssef, Christin Seifert, and Jörg Schlötterer. 2024. LLMs for generating and evaluating counterfactuals: A comprehensive study. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14809–14824, Miami, Florida, USA. Association for Computational Linguistics.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. 2024. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, Miami, Florida, USA. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Alexis Ross, Ana Marasović, and Matthew E. Peters. 2021. Explaining NLP models via minimal contrastive editing (MiCE). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3840–3852. Association for Computational Linguistics.

Anirudh Srinivasan and Eunsol Choi. 2022. TyDiP: A dataset for politeness classification in nine typologically diverse languages. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5723–5738, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Konstantinos Thomas, Giorgos Filandrianos, Maria Lymperaiou, Chrysoula Zerva, and Giorgos Stamou. 2024. "i never said that": A dataset, taxonomy and baselines on response clarity classification. *Preprint*, arXiv:2409.13879.

Marcos Treviso, Alexis Ross, Nuno M. Guerreiro, and André Martins. 2023. CREST: A joint framework for rationalization and counterfactual text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15109–15126, Toronto, Canada. Association for Computational Linguistics.

Yongjie Wang, Xiaoqi Qiu, Yu Yue, Xu Guo, Zhiwei Zeng, Yuhong Feng, and Zhiqi Shen. 2024. A survey on natural language counterfactual generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4798–4818, Miami, Florida, USA. Association for Computational Linguistics.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. 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 6707–6723, Online. Association for Computational Linguistics.

Yangtian Zi, Harshitha Menon, and Arjun Guha. 2025. More than a score: Probing the impact of prompt specificity on LLM code generation. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2380–2402, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

A Datasets and Predictors

To evaluate the robustness and versatility of the framework, five text classification datasets are used, covering sentiment analysis, natural language inference, and pragmatic detection across distinct difficulty levels. Table 4 provides a summary.

The **SNLI** corpus (Bowman et al., 2015) consists of 570,000 premise–hypothesis pairs labeled as *Entailment*, *Neutral*, or *Contradiction*; the framework rewrites the hypothesis while holding the premise fixed. We utilized the same 500 instances used in the **FIZLE** framework (Bhattacharjee et al., 2024), which allows for a direct performance comparison between PROCOG's iterative optimization and FIZLE's static prompting approach.

TweetEval (Barbieri et al., 2020) is the sentiment subset of a unified tweet benchmark with

Dataset	Classes	Task	Classifier
SNLI	3	NLI	RoBERTa-large
TweetEval	3	Sentiment	RoBERTa-base
QEvasion	3	Evasion	RoBERTa-base
Sarcasm	2	Sarcasm	RoBERTa-base
TyDiP	2	Politeness	XLM-RoBERTa

Table 4: Datasets and corresponding target classifiers.

three classes (positive, neutral, negative) and inputs averaging around 20 tokens, testing performance on short, lexically noisy text. We run the framework on the entirety of the test set, which is 12,284 instances.

QEvasion (Thomas et al., 2024) consists of question–answer pairs from political interviews labeled as *Clear Reply*, *Ambivalent*, or *Clear Non-Reply*; because classification depends on the pragmatic coherence between both fields jointly, only the answer is rewritten. We use the 308 instances found in the test set.

Sarcasm Headlines (Misra and Arora, 2023) is a binary benchmark of headlines from *The Onion* (sarcastic) and *HuffPost* (non-sarcastic), where irony is carried by semantic incongruity rather than explicit keywords. Our study utilized the complete test set of 13,355 instances.

TyDiP (Srinivasan and Choi, 2022) provides binary politeness labels across nine languages; we use the English subset, where polite intent is conveyed through subtle prosodic markers rather than lexical sentiment. Our experiments use the English subset, utilizing 250 test instances.

Each dataset is paired with a fine-tuned classifier serving as the fixed black-box function f. The framework receives only the predicted label and confidence score from f, with no access to weights, gradients, or internal representations. We use RoBERTa-large for SNLI and RoBERTa-base for TweetEval, QEvasion, and Sarcasm Headlines; TyDiP uses the XLM-RoBERTa predictor provided by Srinivasan and Choi (2022).

B Full Per-Iteration Trajectory Tables

Tables 5–9 report complete per-iteration trajectories (Fluency ↓, Semantic Similarity ↑, LFR ↑, Minimality ↓) for all six models across the five datasets. Fluency is a relative T5-based score normalized to the source text; Minimality is normalized word-level Levenshtein distance. Claude 3 exhibits severe fluency degradation on Twitter (reaching 15.5 at Iteration 2) before self-correcting. GPT-20b collapses on TyDiP from Iteration 1 onwards (fluency $\geq$ 14.8, similarity $\leq$ 0.34), illustrating the minimum capability threshold required for stable agentic optimization.

Model	It.	Flu. ↓	Sim. ↑	LFR ↑	Min. ↓
Claude 3	0	0.853	0.722	0.101	2.192
	1	0.948	0.686	0.188	2.855
	2	0.873	0.702	0.153	2.665
	3	1.073	0.694	0.198	2.583
	4	1.184	0.696	0.146	2.910
Claude 4	0	1.391	0.708	0.195	3.875
	1	1.013	0.675	0.370	2.952
	2	1.033	0.653	0.461	2.767
	3	1.179	0.665	**0.471**	3.184
	4	0.898	0.653	0.299	2.611
Claude 4.5	0	1.489	0.717	0.166	3.617
	1	3.176	0.747	0.198	4.642
	2	2.217	0.758	0.185	4.091
	3	3.024	0.774	0.156	4.708
	4	3.041	0.780	0.146	4.761
GPT-120b	0	1.085	0.636	0.360	1.920
	1	1.209	0.636	0.253	1.834
	2	0.947	0.627	0.244	1.926
	3	0.922	0.668	0.143	1.857
	4	0.921	0.644	0.133	1.497
GPT-20b	0	1.097	0.690	0.234	1.846
	1	0.911	0.783	0.120	1.445
	2	0.871	0.777	0.130	1.452
	3	0.921	0.778	0.101	**1.342**
	4	0.933	0.782	0.107	1.435
Mistral-L	0	0.658	0.795	0.123	1.784
	1	**0.630**	0.815	0.143	2.227
	2	0.806	0.809	0.159	2.553
	3	0.791	0.812	0.146	2.907
	4	0.647	**0.847**	0.120	1.744

Table 5: Full per-iteration results for **QEvasion** (all 6 models). Claude 4 achieves a 2.4× relative LFR gain over its Iteration 0 baseline (+27.6 pp).

C Dataset Vulnerability Analysis

Results reveal a consistent vulnerability gradient that maps onto the structural properties of each task (Table 1).

High-vulnerability datasets. SNLI and Twitter Sentiment yield the highest LFRs across all models (up to 0.960 and 0.947). Both share short inputs (Twitter ≈20 tokens; SNLI hypotheses ≈10–15 tokens) and classification signals concentrated in surface-level lexical features.

Moderate-vulnerability datasets. TyDiP and Sarcasm Headlines yield mid-range LFRs (best: 0.844 and 0.719). Politeness and sarcasm are conveyed through distributional stylistic patterns, making perturbation harder but not intractable for capable instruction-following models.

High-resistance dataset. QEvasion proves most challenging, with a ceiling of **LFR = 0.471** (Claude 4, Iteration 3). Classification depends

Model	It.	Flu. ↓	Sim. ↑	LFR ↑	Min. ↓
Claude 3	0	0.778	0.424	0.803	1.923
	1	12.779	0.377	0.641	10.227
	2	15.540	0.390	0.626	11.191
	3	0.787	0.353	0.816	2.574
	4	0.808	0.375	0.822	2.326
Claude 4	0	0.759	0.592	**0.947**	1.365
	1	0.745	0.682	0.902	1.565
	2	0.814	0.613	0.871	1.692
	3	0.819	0.612	0.880	1.958
	4	0.818	0.612	0.837	1.811
Claude 4.5	0	0.862	0.466	0.944	1.477
	1	0.820	0.671	0.824	1.233
	2	0.776	0.681	0.851	1.419
	3	0.848	0.688	0.741	1.373
	4	0.869	0.671	0.771	1.390
GPT-120b	0	0.858	0.508	0.640	1.185
	1	0.850	0.511	0.647	1.181
	2	0.841	0.509	0.642	1.184
	3	**0.731**	0.691	0.783	2.154
	4	0.739	0.692	0.784	2.134
GPT-20b	0	0.784	0.575	0.636	1.116
	1	0.811	0.576	0.628	1.126
	2	0.798	0.576	0.628	**1.110**
	3	0.797	0.575	0.629	1.152
	4	0.838	**0.731**	0.537	1.574
Mistral-L	0	5.476	0.530	0.574	4.195
	1	5.683	0.527	0.578	4.356
	2	1.587	0.576	0.751	2.220
	3	1.155	0.590	0.709	1.829
	4	1.208	0.587	0.709	1.919
CAT/BAE-R	–	—	0.871	**0.965**	—

Table 6: Full per-iteration results for **TweetEval** (all 6 models) with CAT/BAE-R baseline (Luu and Inoue, 2023). Claude 3 exhibits severe fluency spikes at Iterations 1–2 before self-correcting at Iteration 3.

Model	It.	Flu. ↓	Sim. ↑	LFR ↑	Min. ↓
Claude 3	0	1.070	0.452	0.329	2.011
	1	1.474	0.497	0.397	1.570
	2	1.380	0.457	0.471	1.528
	3	1.371	0.499	0.354	3.321
	4	1.339	0.498	0.357	3.273
Claude 4	0	0.972	0.554	0.480	2.075
	1	1.118	0.435	0.711	3.254
	2	1.087	0.423	**0.719**	4.086
	3	1.030	0.454	0.705	3.622
	4	1.053	0.451	0.680	3.000
Claude 4.5	0	1.072	0.589	0.431	1.508
	1	1.437	0.467	0.646	1.644
	2	1.369	0.461	0.698	1.627
	3	1.442	0.498	0.608	1.670
	4	1.453	0.483	0.668	1.713
GPT-120b	0	1.018	0.612	0.331	1.162
	1	0.922	0.757	0.406	1.848
	2	**0.884**	0.746	0.407	2.108
	3	0.937	0.754	0.406	1.848
	4	0.938	0.765	0.406	1.827
GPT-20b	0	1.023	0.615	0.348	**1.149**
	1	0.995	0.772	0.410	2.048
	2	0.993	0.810	0.409	1.748
	3	0.998	0.811	0.409	1.736
	4	1.005	**0.812**	0.409	1.751
Mistral-L	0	7.705	0.436	0.343	8.623
	1	6.486	0.445	0.431	7.965
	2	5.359	0.551	0.413	7.695
	3	5.501	0.522	0.401	6.597
	4	5.412	0.521	0.401	6.652

Table 7: Full per-iteration results for **Sarcasm Headlines** (all 6 models).

on pragmatic coherence across a question–answer pair; PROCOG's single-field rewriting is insufficient to manipulate this joint context. We treat this ceiling as a finding about the task's structure and discuss it in Section 6.1.

D Model Performance Analysis

Performance varies considerably across model architectures during the iterative optimization process. The following paragraphs analyze each model with cross-dataset patterns highlighted where applicable.

Claude 4.5 consistently achieved the highest LFR across most datasets, peaking at **0.960** on SNLI—a state-of-the-art result. Performance typically improved steadily from Iteration 0 onwards, suggesting high responsiveness to *Meta-Agent* feedback and strong instruction-following fidelity. The model preserved semantic meaning while maintaining relatively low minimality scores.

Claude 4 delivered the most consistent cross-dataset profile, achieving 0.947 on Twitter, 0.908 on SNLI, 0.828 on TyDiP, and 0.719 on Sarcasm. It produced the best QEvasion result (0.471) through sustained, progressive pattern induction visible in Figure 2b. Unlike smaller models, Claude 4 showed no significant fluency degradation in later iterations, indicating an ability to handle complex constraints without sacrificing linguistic quality.

GPT-120b achieved strong zero-shot performance on SNLI (LFR = 0.926 at Iteration 0) but declined substantially under further optimization, a prime instance of the Specificity Paradox discussed in Section 6.2. On Twitter, it self-improved from 0.640 at Iteration 0 to 0.784 at Iteration 4, confirming that iterative optimization can unlock meaningful gains even for capable zero-shot models when the task provides sufficient signal.

Mistral-Large performed competitively on TyDiP (0.764) and Twitter (0.751) but lagged on SNLI (0.594). Notably, it exhibited a self-correcting trajectory on Twitter, recovering from an initial fluency of 5.476 to 1.155 by Iteration 3, demonstrating that the *Meta-Agent* can partially compensate for early instability, though this behavior was not consistent across datasets.

Claude 3 achieved competitive LFR on simpler datasets (0.822 Twitter, 0.748 TyDiP) but exhibited severe fluency spikes during optimization on

Model	It.	Flu. ↓	Sim. ↑	LFR ↑	Min. ↓
PROCOG *(this work)*					
Claude 3	0	0.410	0.355	0.666	2.137
	1	0.517	0.334	0.648	4.238
	2	0.530	0.310	0.656	4.029
	3	0.524	0.324	0.632	4.357
	4	0.517	0.321	0.644	4.308
Claude 4	0	0.393	0.319	0.788	**1.349**
	1	0.399	0.278	0.908	1.590
	2	0.414	0.295	0.886	1.635
	3	0.405	0.420	0.774	1.528
	4	0.402	0.380	0.808	1.562
Claude 4.5	0	0.397	0.306	0.920	1.353
	1	0.468	0.376	**0.960**	1.376
	2	0.440	0.371	0.944	1.463
	3	0.459	0.405	0.934	1.521
	4	0.455	0.406	0.936	1.515
GPT-120b	0	0.428	0.320	0.926	1.523
	1	**0.380**	0.624	0.642	1.624
	2	0.404	0.465	0.582	1.689
	3	0.432	0.353	0.586	1.548
	4	0.421	0.285	0.650	1.766
GPT-20b	0	0.431	0.325	0.874	1.465
	1	0.401	0.301	0.856	1.491
	2	0.434	0.629	0.496	1.886
	3	0.552	0.132	0.642	2.300
	4	0.401	**0.655**	0.472	1.873
Mistral-L	0	0.542	0.371	0.590	2.543
	1	1.869	0.368	0.580	7.790
	2	1.985	0.354	0.594	9.005
	3	1.257	0.476	0.514	6.349
	4	2.318	0.455	0.548	9.797
FIZLE *(Bhattacharjee et al., 2024)*					
GPT-3.5	N	0.471	**0.918**	0.520	0.605
	G	0.371	0.840	0.350	0.821
GPT-4	N	0.372	0.873	0.792	0.773
	G	0.589	0.751	0.800	0.745
GPT-4o	N	0.344	0.883	0.764	0.755
	G	0.354	0.909	0.658	0.852
GPT-4o-mini	N	0.387	0.819	0.484	0.591
	G	0.362	0.753	0.473	0.727
Llama 3 8B	N	0.406	0.748	0.578	0.262
	G	**0.309**	0.789	0.531	**0.775**

Table 8: Full per-iteration results for **SNLI** (all 6 PROCOG models) with all FIZLE baselines (Bhattacharjee et al., 2024). It. = iteration; N = Naive, G = Guided prompting strategy.

Twitter (reaching 15.5 at Iteration 2) before self-correcting, indicating difficulty maintaining textual coherence under aggressive constraint injection.

GPT-20b displayed a qualitatively distinct and practically significant failure mode: its extended CoT sequences frequently exceeded the 4,000-token generation limit, causing the refined prompt to be truncated. When no valid new prompt could be extracted, the system retained the previous iteration's prompt—producing apparently identical metrics across consecutive iterations in QEvasion and TyDiP. This is not model convergence; it is a silent infrastructure failure. On TyDiP, a single malformed prompt entering the loop caused fluency scores to spike catastrophically above 18.0. This

Model	It.	Flu. ↓	Sim. ↑	LFR ↑	Min. ↓
Claude 3	0	0.461	0.543	0.404	1.748
	1	0.482	0.406	0.748	2.056
	2	0.530	0.429	0.668	2.480
	3	0.458	0.412	0.732	2.501
	4	0.482	0.449	0.668	2.563
Claude 4	0	0.480	0.546	0.824	1.686
	1	0.480	0.615	0.828	1.857
	2	0.485	0.625	0.816	1.928
	3	0.516	0.613	0.804	1.929
	4	0.493	0.621	0.808	1.943
Claude 4.5	0	0.551	0.455	0.832	1.672
	1	0.509	0.567	**0.844**	2.264
	2	0.616	0.566	0.836	2.265
	3	0.537	0.571	0.836	2.249
	4	0.595	0.574	**0.844**	2.282
GPT-120b	0	**0.437**	0.709	0.360	0.855
	1	0.746	0.745	0.488	0.974
	2	0.732	0.692	0.660	1.175
	3	0.784	0.705	0.628	1.123
	4	0.721	0.697	0.580	1.072
GPT-20b	0	0.445	0.751	0.268	**0.747**
	1	14.815	0.188	0.332	8.400
	2	18.356	0.336	0.360	32.128
	3	18.950	0.122	0.384	12.524
	4	18.506	0.169	0.292	13.924
Mistral-L	0	5.291	0.550	0.412	3.498
	1	0.458	0.761	0.764	1.068
	2	0.466	0.748	0.764	1.070
	3	0.468	**0.765**	0.700	1.003
	4	0.441	0.758	0.708	1.004

Table 9: Full per-iteration results for **TyDiP** (all 6 models). GPT-20b undergoes catastrophic degradation from Iteration 1 (fluency reaching 18.95, similarity dropping to 0.12 at Iteration 3).

finding has direct implications for agentic system design: reasoning models with extended thinking capabilities require explicit token budget management and output format validation to function reliably in iterative pipelines.

E Label Enforcement Examples

Label Enforcement (Section 6.3) occurs when the *Generator* produces a counterfactual that preserves or reinforces the original label rather than flipping it. The mechanism, as described in Section 6.3, is that a prompt enumerating all target classes in detail causes the LLM to match the input to the most similar class description and generate conforming text, overriding the flip instruction. The following examples illustrate this failure mode across four datasets where it manifests most clearly. Failures on QEvasion are better characterized as classifier bias rather than Label Enforcement, and are discussed separately in Section 6.4.

SNLI (Textual Entailment). Entailment enforcement occurs when the *Generator* paraphrases the premise directly into the hypothesis, inadvertently strengthening the entailment relation rather than

inverting it. The model recognizes the entailment class description ("text that restates or follows from the premise") and generates a conforming output.

> *Premise*: "Three people sit on a bench at a station, the man looks oddly at the two women. . . "
> *Original hypothesis* (Entailment, 87.6%): "People wait at a station."
> *Generated hypothesis* (Entailment, 99.1%): "There are three people sitting on a bench."

The *Generator* restated the premise using the standard entailment template ("There are. . . "), reinforcing the label it was instructed to flip and pushing classifier confidence to 99.1%.

TweetEval (Twitter Sentiment). The TweetEval example is particularly striking because the original tweet expresses only mild, hedged positivity. Rather than exploiting this ambiguity to produce a neutral or negative rewrite, the *Generator* defaulted to a prototypical positive rewrite, amplifying the very signal it was instructed to invert.

> *Original* (Positive, 82.3%): "So that Westworld show, it isn't too bad. I guess. If you like that sort of thing."
> *Generated* (Positive, 98.9%): "So that Westworld show is absolutely incredible! I'm obsessed and can't wait for the next episode to air this Sunday!"

The *Generator* amplified the sentiment signal from cautious approval to enthusiastic endorsement, increasing classifier confidence by +16.6 pp. This illustrates how Label Enforcement can arise not from a failure to understand the instruction, but from the model defaulting to the most prototypical version of the source label.

Sarcasm Headlines. On Sarcasm Headlines, Label Enforcement arises when the *Generator* matches the satirical register of the original headline and produces a new sarcastic output rather than a neutral one, often amplifying the very stylistic markers that drive the classifier's prediction.

> *Original* (Sarcastic, 95.9%): "economists recommend setting aside part of every paycheck in case of dire straits reunion tour"
> *Generated* (Sarcastic, 98.8%): "Area man desperately liquidating retirement fund after learning favorite 80s rock band planning comeback tour. . . "

The model produced a new sarcastic headline with a different setup but the same satirical framing, increasing classifier confidence by +2.9 pp.

TyDiP (Politeness Classification). On TyDiP, a particularly clear instance of Label Enforcement occurs on a near-boundary example that a human reader would likely perceive as sarcastic and borderline impolite. Rather than exploiting this ambiguity to produce an impolite rewrite, the *Generator* recognized the polite class description and generated a prototypically courteous output, amplifying the polite signal instead of inverting it.

> *Original* (Polite, 79.7%): "I'm sure. How about we let the sockpuppet investigation let us know for certain, m'kay?"
> *Generated* (Polite, 99.7%): "I appreciate your perspective. Perhaps we could wait for the sockpuppet investigation to provide clarity before taking further action. Would that work for you?"

The original text carries a dismissive, sarcastic tone ("m'kay?") that human annotators would likely classify as impolite, yet the classifier assigns it a polite label at 79.7% confidence. Rather than exploiting this near-boundary ambiguity, the *Generator* stripped the sarcasm entirely and produced a formally courteous rewrite, increasing classifier confidence by +20.0 pp. This example also surfaces a classifier bias: the divergence between the model's polite label and the human-perceived sarcastic tone suggests the classifier is matching surface politeness markers rather than pragmatic intent—a finding consistent with the token-level bias documented in Section 6.4.

F Foundation Models and Reproducibility

Implementation Details. All experiments run for five iterations (0–4), with Iteration 0 serving as the task-agnostic few-shot baseline. In-context examples are selected via K-means clustering over `all-MiniLM-L6-v2` embeddings from the `sentence-transformers` library, with two examples per class. The *Generator* and *Meta-Agent* use temperatures of 0.7 and 0.3, respectively. The *Meta-Agent* is limited to 4,000 output tokens; if prompt refinement fails due to truncation, the previous prompt is retained. Early stopping is disabled to preserve full optimization trajectories.

Tables 10–11 provide model cards for all models used in PROCOG. *Generator* and *Meta-Agent* roles are fulfilled by the same model unless otherwise specified. All *Generator* and *Meta-Agent* models are accessed via AWS Bedrock (us-east-1). Predictor models are accessed via the Hugging Face Hub or local filesystem; the framework receives only

predicted labels and confidence scores from each predictor, with no access to weights, gradients, or internal representations.

G Prompts

We present the initial task-agnostic prompt (P_0) and two representative refined prompts produced by the *Meta-Agent* for Claude 4 on SNLI, illustrating the Specificity Paradox. Table 12 summarises the prompt evolution across all five iterations.

The *Meta-Agent* prompt that drives refinement between iterations follows the four-step protocol described in Section 4.2: Pattern Induction, Instruction Debugging, Constraint Synthesis, and Template Generation. It receives the current prompt P_t, structured failure counts and examples per failure type, and the full optimization history, and operates at temperature $T = 0.3$ within a 4,000-token output budget.

H Counterfactual Examples

Tables 13–16 present curated counterfactual examples from the SNLI and QEvasion runs with Claude 4. Table 15 traces a single SNLI instance across all five iterations to illustrate how prompt refinement progressively corrects a failed generation.

I Reproducibility

Infrastructure. All LLM-based counterfactual generation and *Meta-Agent* prompt refinement runs were executed on Amazon Bedrock (region: `us-east-1`; models with `us.` prefix use cross-region inference). Task-specific classifiers and the semantic similarity model (`all-MiniLM-L6-v2`) ran locally on CPU/GPU (`device: auto`).

Hyperparameters. All experiments run for five iterations (0–4) with random seed 42. In-context examples are selected via K-Means clustering with two examples per class. The *Generator* operates at temperature $T = 0.7$; the *Meta-Agent* at $T = 0.3$ within a 4,000-token output budget. Early stopping is disabled to allow full trajectory observation. Table 17 summarises the full experimental setup.

Runtimes. Table 18 reports the total optimization loop runtime (generate–evaluate–refine) for each dataset–model combination, excluding initial model loading and I/O. Runtimes scale with the number of test instances and input length: QEvasion question–answer pairs are substantially longer than SNLI hypotheses, explaining the disproportionate cost per instance. GPT-20b is notably slower than all other models due to its extended chain-of-thought generation, with TweetEval requiring over 129 hours for a single full run.

Software. LLM inference used the AWS `boto3` SDK (Bedrock Runtime, `invoke_model`). Classifiers and embeddings used Hugging Face `transformers` and `sentence-transformers`. Datasets were loaded via the Hugging Face `datasets` library.

Data sources. SNLI few-shot examples were drawn from the `stanfordnlp/snli` training split; the 500-instance test set matches the evaluation split of Bhattacharjee et al. (2024) for direct comparability. QEvasion and TweetEval were loaded from Hugging Face (`ailsntua/QEvasion` and `cardiffnlp/tweet_eval`). Sarcasm Headlines and TyDiP were loaded from local files.

Output artifacts. Optimization outputs are saved as JSON files under `outputs/<task>/<model>/` and contain the full run configuration, per-iteration prompts and metrics, individual counterfactual results, and runtime metadata. These artifacts will be released alongside the code upon publication.

Model	Model ID	Context	Max Out	Provider
Claude 4	`claude-sonnet-4-20250514-v1:0`	200K	64K	Anthropic
Claude 4.5	`claude-sonnet-4-5-20250929-v1:0`	1M	64K	Anthropic
Claude 3	`claude-3-sonnet-20240229-v1:0`	200K	4K	Anthropic
GPT-120b	`openai.gpt-oss-120b-1:0`	—	—	OpenAI
GPT-20b	`openai.gpt-oss-20b-1:0`	—	—	OpenAI
Mistral-Large	`mistral-large-2402-v1:0`	32K	—	Mistral AI

Table 10: *Generator* and *Meta-Agent* models. All accessed via AWS Bedrock (us-east-1). Parameter counts are not publicly disclosed. GPT-20b uses extended chain-of-thought reasoning; outputs may contain reasoning traces requiring post-processing.

Dataset	Model ID	Architecture	Source
SNLI	`ynie/roberta-large-snli_mnli`	RoBERTa-Large	HF Hub
TweetEval	`cardiffnlp/twitter-roberta-base`	RoBERTa-Base	HF Hub
QEvasion	`roberta_model_clarity`	RoBERTa-Base	Local
Sarcasm	`helinivan/english-sarcasm-detector`	BERT-Base	HF Hub
TyDiP	`Genius1237/xlm-roberta-large-tydip`	XLM-RoBERTa-Large	HF Hub

Table 11: Target classifiers used as black-box predictors f. Full model IDs are abbreviated; complete identifiers are in the released code.

It.	Key change	LFR
0	Generic: study examples, infer classification basis	0.788
1	+ Task definition (textual entailment), label descriptions, transformation strategies	**0.908**
2	Minor wording refinements	0.886
3	+ Highly specific patterns (e.g., "Man with bike → Man and a bike"), critical success factors	0.774
4	Same as Iteration 3	0.808

Table 12: Prompt evolution across iterations for Claude 4 on SNLI. LFR peaks at Iteration 1 then declines as specificity increases, illustrating the Specificity Paradox (Section 6.2).

Generator Prompt P_0 (Iteration 0 — Task-Agnostic Baseline)

Your task is to generate counterfactual versions of text that would be classified as a different label by a machine learning model. These are different example instances of the dataset, where each label is represented, grouped by the classification label.
`{examples}`
From the examples, try to discern what distinguishes different example types from each other. Find the basis for the classification, and the parts of the text that are used to make the classification.
If the examples show two text fields, note that you should ONLY rewrite the second text portion. The first text must remain unchanged.
Guidelines for rewriting:
1. Carefully study the provided examples to understand the dataset.
2. Identify what distinguishes different example types from each other.
3. Apply the patterns you observe to create the rewritten version.
4. The goal is to change the label classification—make bold changes if needed.
5. Maintain natural, coherent language; keep the generated text under 100 words.
6. Focus on changing the key elements that determine the classification label.
7. Don't worry about similarity to the original—prioritize label change.
CRITICAL OUTPUT FORMAT: Output ONLY the rewritten text. No prefixes, no headers, no labels, no explanations.
This is the original text you have to change: `{original_text}`
Counterfactual:

Generator Prompt P_1 (Iteration 1 — Peak LFR 0.908)

Your task is to generate a counterfactual version of text that will be classified as a DIFFERENT label by a machine learning model. This is a textual entailment task with three categories: entailment (Label 0), neutral (Label 1), and contradiction (Label 2).
`{examples}`
UNDERSTANDING THE TASK:
- Label 0 (Entailment): The second text is directly supported by or logically follows from the first text.
- Label 1 (Neutral): The second text makes reasonable inferences or adds plausible information not directly stated in the first text.
- Label 2 (Contradiction): The second text directly contradicts factual elements in the first text.

TRANSFORMATION STRATEGIES:
To create Label 0: Summarise or paraphrase key facts from the first text. Use "There are...", "A person is...", "The scene shows..."
To create Label 1: Add reasonable context or backstory not stated in the original. Use "learning how to...", "trying to...", "enjoying..."
To create Label 2: Change key factual elements—gender, number of people, objects, locations, actions.
REWRITING INSTRUCTIONS:
1. Identify the current label category of the original text.
2. Choose a DIFFERENT target label category.
3. Rewrite the second text ONLY to fit the target category.
4. Make bold, clear changes—subtle modifications often fail to flip labels.
5. Keep the rewritten text natural and under 100 words.

CRITICAL OUTPUT: Output ONLY the rewritten second text. No prefixes, no explanations, no labels.
Original text to transform: `{original_text}`

Generator Prompt P_3 (Iteration 3 — Specificity Paradox)

[Same task definition and label descriptions as P_1.]
TRANSFORMATION STRATEGIES WITH SPECIFIC PATTERNS:
To create Label 0: Start with "There are..." or "A person is...". Directly restate key nouns and actions: "Man with bike" → "Man and a bike". Use present-tense statements of visible facts only.
To create Label 1: Add emotional or intentional context. Make reasonable assumptions about purpose. Use "might be", "appears to be", "seems like".
To create Label 2 (exact patterns): Change gender (man↔woman), change numbers ("two people" → "one person"), change actions ("sitting" → "standing"), change objects ("dog" → "cat"), change locations completely.
CRITICAL SUCCESS FACTORS:
- For failed label flips: make MORE dramatic changes to core elements.
- For entailments, stick closely to explicitly stated facts only.
- For inferences, add context that could be true but is not stated.

Original text to transform: `{original_text}`

Original hypothesis	**Counterfactual**	**Labels**	**Sim**
The church is filled with song	The church is completely silent and empty	ent.→con.	0.56
The woman is young	The woman is elderly with a frown and wears a red hat and yellow dress	neu.→con.	0.37
The woman has been shot	The woman is smiling happily	con.→neu.	0.30
A sedan is stuck in the middle of a river	A land rover is being driven across a river	con.→ent.	0.47
A vehicle is crossing a river	A vehicle is parked in a garage	ent.→con.	0.30
There is a statue that not many people seem interested in	A crowd of visitors eagerly gathered around examining the statue	ent.→con.	0.62

Table 13: Successful counterfactual flips on SNLI (Claude 4, Iteration 1). Labels: ent. = entailment, neu. = neutral, con. = contradiction. Sim = cosine similarity between `all-MiniLM-L6-v2` embeddings.

Original hypothesis	**Counterfactual**	**Failure type**	**Sim**
A choir singing at a baseball game	A choir performing silently in an empty desert	Type I (still con.)	0.41
A man walks by an ad	A woman runs away from the advertisement	Type I (wrong strategy)	0.36
A man walks by an ad	A young woman without belongings sits inside a store	Type III (too dissimilar)	−0.04
A blond-haired doctor and her assistant...	A woman is eating pb and j	Type I (gender swap insufficient)	0.75

Table 14: Failed counterfactual flips on SNLI. Type I = label not flipped; Type III = too dissimilar. High similarity (0.75) does not guarantee a label flip; low similarity (−0.04) indicates a semantically unrelated generation.

It.	Counterfactual hypothesis	Flip?	Sim
0	A woman runs away from the advertisement	×	0.36
1	A young woman without belongings sits inside a store	×	−0.04
2	A person is standing with a package in front of an advertisement	✓	0.49
3	A man is standing with a package near an advertisement	✓	0.57
4	A man is standing with a package near an advertisement	✓	0.57

Table 15: Single SNLI instance traced across all five iterations. Premise: *An old man with a package poses in front of an advertisement.* Original hypothesis: *A man walks by an ad* (contradiction). Iterations 0–1 produce failed flips; from Iteration 2 the *Meta-Agent* steers the *Generator* toward premise paraphrase (entailment), achieving consistent success.

Original answer (excerpt)	Counterfactual (excerpt)	Labels	Flip?
Deflective hedging, pivot to broader picture...	Absolutely, the polling numbers show declining support. Just yesterday, three more senators expressed serious concerns...	LABEL_1→LABEL_0	✓
Deflective, general policy statement...	Pardon me? Could you be more specific? I'm not sure I understand the question.	LABEL_1→LABEL_2	✓
The redline status is clear—we have established specific thresholds, three specific criteria: nuclear payload capacity above 50 kilotons...	—	LABEL_1→LABEL_1	×

Table 16: QEvasion counterfactual examples (Claude 4). Labels: LABEL_0 = Direct Reply, LABEL_1 = Deflective, LABEL_2 = Evasive. The failed flip (row 3) illustrates holistic feature reliance: explicit directness markers fail to overcome the classifier's sensitivity to multi-clause sentence structure (Section 6.4).

Parameter	Value	Note
Max iterations	5	Iterations 0–4
Few-shot per label	2	Via K-Means clustering
Generator temp.	0.7	
Meta-Agent temp.	0.3	Analytical consistency
Meta-Agent budget	4,000 tokens	Truncation retains P_t
Early stopping	Disabled	Full trajectory observation
Random seed	42	Splits and example selection
Dataset	**Test instances**	**Source**
SNLI	500	Same split as Bhattacharjee et al. (2024)
QEvasion	308	Full test split
TweetEval	12,284	Full test split
Sarcasm Headlines	13,355	Full test split
TyDiP	250	English subset

Table 17: Experimental setup and dataset instance counts.

Dataset	Claude 3	Claude 4	Claude 4.5	Mistral-L	GPT-120b	GPT-20b
Total runtime (hours)						
SNLI (500)	1.2	1.5	2.4	1.4	2.1	11.3
QEvasion (308)	5.4	27.5	3.8	2.3	25.7	4.6
TweetEval (12,284)	66.8	33.2	56.0	29.0	62.0	129.3
Sarcasm (13,355)	29.7	42.0	51.1	52.0	47.7	49.4
TyDiP (250)	1.1	1.3	1.8	0.7	0.9	5.3
Average per-iteration runtime (minutes)						
SNLI	15	18	28	17	25	135
QEvasion	65	330	47	27	381	62
TweetEval	914	398	672	373	827	1,696
Sarcasm	335	504	613	644	617	750
TyDiP	13	16	22	9	11	70

Table 18: Optimization loop runtimes per dataset–model combination. Instance counts shown in parentheses. All LLM inference ran on Amazon Bedrock (us-east-1); classifiers and similarity models ran locally. GPT-20b's disproportionate cost reflects its extended chain-of-thought generation. QEvasion runtimes for Claude 4 and GPT-120b are elevated due to longer question–answer inputs and more aggressive Meta-Agent refinement cycles.