PRIME: Prompt Refinement via Iterative Mutation and Evolution for Robust LLM Annotation

Anonymous ACL submission

Abstract

Large language models are increasingly used as scalable data annotators, but their quality under domain and user distribution shifts remains highly sensitive to prompt design. We introduce **PRIME**, a prompt-only framework that adapts a frozen LLM ensemble by evolving a shared structured prompt through any model exposed via an inference API. PRIME combines an AlphaEvolve-style LLM mutator with an active-learning controller that repeatedly exposes the mutator to hard or high-disagreement examples, consolidates stable rules, and optionally injects synthetic few-shot boundary examples. On the out-of-distribution (OOD) review dataset, PRIME improves over the LLM ensemble and fine-tuned approaches in both worst-group and global accuracies. Our ablations show that active batch rotation and synthetic boundary examples are key for OOD robustness. The framework is compatible with black-box models accessed through commercial or self-hosted API endpoints.

1 Introduction

Large language models (LLMs) are increasingly used as scalable annotators for NLP datasets (Gilardi et al., 2023; Ding et al., 2023; Tan et al., 2024), offering a practical alternative to manual labeling for large and evolving corpora. However, annotation quality under domain and user distribution shifts remains highly sensitive to prompt design, particularly when models are accessed only through inference APIs and cannot be fine-tuned (Gururangan et al., 2020; Hu et al., 2022). Recent work shows that even minor prompt variations can substantially alter LLM predictions (Sclar et al., 2024; Mizrahi et al., 2024), making prompt optimization a central challenge for robust automatic annotation. In practice, prompt engineering remains largely manual and brittle, requiring repeated human intervention to resolve failure cases and adapt prompts to new data distributions.

Recent approaches to automatic prompt optimization, including evolutionary search, self-refinement, and LLM-based prompt mutation, demonstrate that prompts can be optimized without gradient access or weight updates (Zhou et al., 2023; Pryzant et al., 2023; Yang et al., 2024; Guo et al., 2024). However, most existing methods optimize prompts on static datasets or fixed validation subsets, treating all examples as equally informative. For annotation under distribution shift, this is inefficient: many examples are already easy under the current prompt, while the remaining errors concentrate around ambiguous or unstable boundary cases. Active learning methods address a related problem by selecting informative examples through uncertainty, disagreement, or diversity (Settles, 2009; Ein-Dor et al., 2020; Margatina et al., 2021), and recent work adapts these principles to in-context prompting (Margatina et al., 2023; Diao et al., 2024). These ideas have not yet been tightly coupled with iterative prompt evolution for robust LLM annotation. As a result, prompt search can improve average performance while remaining fragile on hard or out-of-distribution examples (Koh et al., 2021; Kumar et al., 2022).

We introduce PRIME (**P**rompt **R**efinement via **I**terative **M**utation and **E**volution), a prompt-only adaptation framework that combines evolutionary prompt search with active learning for robust LLM annotation. PRIME maintains a shared structured prompt for a frozen ensemble of API-accessible LLM annotators and iteratively refines this prompt using an AlphaEvolve-style mutator guided by informative failures and inter-model disagreement. The prompt is explicitly partitioned into stable task rules, mutable cycle-level rules, and few-shot examples, so the mutator can revise the parts that are meant to change without rewriting the entire instruction. This reduces variance across mutation steps and preserves useful rules that would otherwise be overwritten by monolithic prompt edits. On

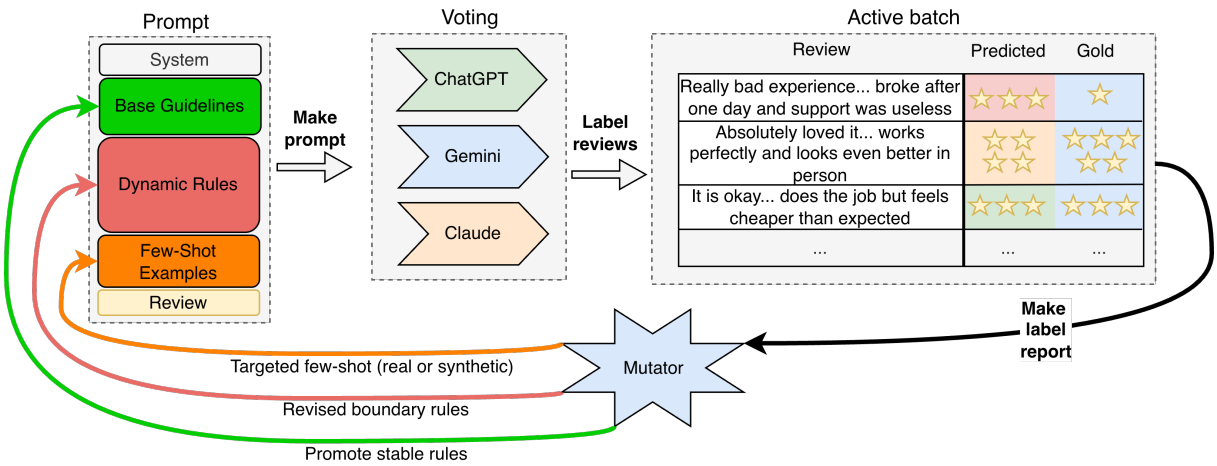

Figure 1: Solution architecture for Prompt Refinement via Iterative Mutation and Evolution. The current structured prompt is instantiated for each review and sent to a LLM ensemble. Majority voting produces predicted labels for the active batch, which are compared against gold labels to form a label report. The mutator then revises the prompt through three update paths: targeted few-shot examples (real or synthetic), revised boundary rules, and promoted stable rules.

Amazon-WILDS, a benchmark for sentiment classification under user distribution shift, PRIME improves worst-group accuracy from 50.0% to 55.9% and outperforms strong fine-tuning baselines while requiring no weight updates.

Contributions. This paper makes three main contributions:

1. We couple evolutionary prompt search with active learning for annotation under distribution shift, using an AlphaEvolve-based mutator and dynamic Hard/Anchor batches driven by majority-vote errors and worker disagreement.

2. We introduce a structured prompt representation with separate within-cycle and between-cycle update rules, limiting catastrophic forgetting in prompt space.

3. We demonstrate strong results on Amazon WILDS and ablate the contributions of active learning and synthetic few-shot examples.

2 Related Work

Active learning for LLM annotation. Crowd and expert labeling remains the traditional standard for NLP annotation, but it is expensive, slow, and sensitive to annotator disagreement (Snow et al., 2008; Sabou et al., 2014; Klie et al., 2024). LLM annotation scales better, but generated labels still require validation and domain-aware quality control (Gilardi et al., 2023; Wang et al., 2021; Ding et al., 2023; Tan et al., 2024). Active learning provides a natural mechanism for reducing annotation effort by focusing on examples that are likely to be informative: classical acquisition functions include uncertainty sampling, query-by-committee disagreement, expected model change, and diversity-aware sampling (Settles, 2009). For modern text models, empirical studies show that uncertainty and representation diversity are useful signals for BERT-style classifiers (Ein-Dor et al., 2020), and contrastive acquisition can select boundary examples near competing classes (Margatina et al., 2021). Recent work brings active selection to prompting by choosing demonstrations for in-context learning or chain-of-thought prompting (Margatina et al., 2023; Diao et al., 2024). PRIME uses the same principle for prompt evolution: majority-vote errors and inter-worker disagreement define Hard examples, while clustering keeps the active batch diverse and Anchor examples guard against regressions.

Distribution-shift baselines and robust fine-tuning. Distribution shift methods aim to avoid models that fit the average training distribution while failing on minority or unseen domains. WILDS formalizes this setting through real-world train/test splits and tail metrics, including Amazon-WILDS, where domains are reviewers, labels are 1-to-5 ratings, and train and test users are disjoint (Koh et al., 2021). Standard ERM minimizes average loss and can overfit to majority patterns,

so robust baselines modify the training objective or data weighting. GroupDRO directly upweights high-loss groups to optimize worst-group performance (Sagawa et al., 2020); IRM encourages invariant predictors across environments (Arjovsky et al., 2019); JTT first identifies high-loss examples and then upweights them in a second training stage (Liu et al., 2021); and LISA improves OOD robustness through selective mixup-style augmentation across domains (Yao et al., 2022). Related work also shows that simple balancing can be competitive for worst-group accuracy (Idrissi et al., 2022), while naive fine-tuning may distort pretrained features and underperform OOD (Kumar et al., 2022). These approaches all update model weights, usually with a pretrained LLM backbone, whereas PRIME targets black-box setting where only inference APIs and prompt text are available.

Automatic prompt optimization and structured prompts. Prompt sensitivity motivates automatic prompt optimization because small wording or formatting changes can alter LLM behavior enough to make single-prompt evaluation unreliable (Sclar et al., 2024; Mizrahi et al., 2024). APE, APO, OPRO, and DSPy optimize prompts with LLM-generated candidates, textual feedback, or declarative compilation (Zhou et al., 2023; Pryzant et al., 2023; Yang et al., 2024; Khattab et al., 2024), while PromptBreeder and EvoPrompt use evolutionary operators over prompt populations (Fernando et al., 2024; Guo et al., 2024). A parallel line uses LLMs as mutators in program search, with FunSearch combining LLM mutation, large-scale evaluation, and diversity-preserving archives (Romera-Paredes et al., 2024). PRIME follows this evolutionary paradigm but makes the prompt representation structured: stable task guidelines, mutable boundary rules, and few-shot examples are updated through separate paths. This design constrains the search space and reduces destructive edits, which is important when prompt improvements must accumulate across active-learning cycles rather than merely win on a fixed validation subset.

3 Method

3.1 Framework Overview

We propose **Prompt Refinement via Iterative Mutation and Evolution (PRIME)**, a framework that adapts an ensemble of frozen LLMs to a specific domain solely by evolving their shared instruction set (the prompt). The framework operates on two nested levels. The inner loop uses OpenEvolve[1] (Sharma, 2025), an open-source implementation of the AlphaEvolve paradigm, where an LLM acts as a mutator to iteratively refine prompt candidates against an active batch of training data. The outer loop implements an active-learning strategy: it evaluates the current best prompts, consolidates their rules, and dynamically updates the active batch by sampling the most informative examples from an unlabeled pool. Figure 1 summarizes the architecture.

3.2 Problem Formulation

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote a labeled dataset of reviews (or, more generally, texts) with ordinal labels $y_i \in \{1, \ldots, 5\}$. We assume access to a fixed ensemble of M annotator LLMs $\{f_m\}_{m=1}^{M}$, each invoked with a shared natural-language prompt P. The ensemble prediction is obtained by majority vote:

$$\hat{y}(x; P) = \mathrm{mode}\big(\{f_m(x; P)\}_{m=1}^{M}\big). \quad (1)$$

A prompt P is represented as a structured document (Section 3.3) rather than a monolithic string. Our goal is to find a prompt that maximizes annotation quality under distribution shift while keeping all f_m frozen:

$$P^{\star} \in \arg\max_{P \in \mathcal{P}} \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{test}}} \left[\mathbf{1}\{\hat{y}(x; P) = y\}\right], \quad (2)$$

where $\mathcal{P}$ is the space of valid structured prompts and $\mathcal{D}_{\text{test}}$ follows an out-of-distribution (OOD) user split (Section 4). Because $\mathcal{P}$ is discrete and non-differentiable, we optimize P with a genetic search guided by a scalar fitness $\mathcal{F}(P)$ (Section 3.6) and an active-learning controller that selects which examples enter each evolutionary episode.

3.3 Structured Prompt Representation

Monolithic prompts are fragile under LLM-driven mutation: fixing a new error can delete a rule that previously worked, a form of *catastrophic forgetting in prompt space*. Because prompt formatting and wording can strongly affect LLM predictions (Sclar et al., 2024; Mizrahi et al., 2024), unconstrained rewrites also add variance that is difficult to attribute to a specific rule change. We therefore expose only selected prompt sections to each

[1] https://github.com/algorithmicsuperintelligence/openevolve

update stage and keep stable instructions separate from exploratory edits. We therefore decompose P into four interpretable sections:

<System> Role definition and a concise task description for the annotator ensemble.

<BaseGuidelines> Stable rules promoted from successful evolutionary runs. Updated only during *consolidation* (Section 3.7).

<DynamicRules> Experimental hypotheses refined by the mutator during the inner loop. This is the primary search space within each active-learning cycle.

<FewShotExamples> Few-shot demonstrations, refreshed to cover current confusion patterns and boundary cases.

Two-level update policy.
Within-cycle (Level 1): during each outer AL cycle, the mutator may edit <DynamicRules> and <FewShotExamples> only (Algorithm 1, Level 1).
Between-cycle (Level 2): after a cycle ends, consolidation may merge proven rules into <BaseGuidelines> (Section 3.7).

Level 1 enables fast adaptation to the current active batch; Level 2 promotes only durable rules into the long-term prompt backbone. This separation preserves accumulated domain knowledge while still allowing rapid adaptation to newly acquired hard examples.

3.4 Synthetic Few-Shot Generator

When enabled, an auxiliary LLM call runs before Level 1 evolution (Algorithm 1, optional step) to provide controlled minimal-pair contrasts for dominant error boundaries. The generator reads top confusion pairs (e.g., gold rating 3 → predicted rating 4), misclassified examples, and current <DynamicRules>, then creates synthetic *twin* review pairs that differ in one rating-relevant cue while covering the 1–5 scale. The resulting examples are either injected as a mutator hint or used to refresh <FewShotExamples>. Main experiments keep this module enabled; Section E ablates it, and Appendix F shows a concrete example set. The motivation is that many Amazon errors are adjacent-class errors rather than arbitrary label flips: reviews with mixed praise and criticism are often confused between 3 and 4 stars, while severe product failures may be confused between 1 and 2 stars. Synthetic twin pairs make these local boundaries explicit by holding most review content fixed and changing only one rating-relevant cue. This gives the mutator a compact description of what should change the label without adding noisy real examples to the active batch.

3.5 Active Learning Loop and Batch Construction

Evaluating every prompt candidate on the full training pool at each mutation is prohibitively expensive. Instead, we maintain a dynamic **active batch** $\mathcal{B}_t$ of size $|\mathcal{B}_t| = B$ (e.g., B=80), rebuilt at the start of each outer cycle t. Prompt discovery starts from a compact but diverse seed training pool sampled via clustering from the full WILDS training split; during active learning, this pool can be expanded with additional examples from the full training split whenever the controller needs new hard cases. The validation and development test splits used for cycle-level monitoring are constructed representatively across product categories (Appendix D).

Seen / Unseen pool. The training pool is partitioned into **seen** examples $\mathcal{S}_t$ already evaluated at least once and **unseen** examples $\mathcal{U}_t$ not yet used for adaptation; initially, $\mathcal{S}_0 = \emptyset$ and $\mathcal{U}_0 = \{1, \ldots, N\}$.

Hard / Anchor split. After evaluation with the current prompt, each seen example $i \in \mathcal{S}_t$ is labeled using ensemble outputs $\{f_m(x_i; P)\}_{m=1}^{M}$:

$$d_i = \frac{1}{\binom{M}{2}} \sum_{1 \le a < b \le M} \left| f_a(x_i; P) - f_b(x_i; P) \right|, \quad (3)$$

where $y_{\min}$=1 and $y_{\max}$=5, so d_i is normalized to $[0, 1]$ over the rating scale. Example i is assigned to the **Hard** set $\mathcal{H}_t$ if the ensemble majority vote is incorrect ($\hat{y}(x_i; P) \neq y_i$) or worker disagreement exceeds a threshold ($d_i > \tau$), and to the **Anchor** set $\mathcal{A}_t$ otherwise. In our reported runs we use τ=0 (Appendix A), meaning that any worker-level disagreement is treated as uncertain even if the majority vote is correct. Intuitively, $\mathcal{H}_t$ contains examples the ensemble majority misclassifies or examples on which workers disagree; $\mathcal{A}_t$ contains examples the current prompt handles correctly and consistently.

Batch assembly. The active batch is assembled with a fixed quota of Hard and Anchor examples (e.g., 70% Hard and 30% Anchor, with ρ=0.7). We use B=80 and ρ=0.7 in all reported runs: enough

Algorithm 1 PRIME: one outer cycle

Require: Seed prompt P_t; active batch size B; unseen pool $\mathcal{U}_t$
Ensure: Updated prompt P_{t+1}
Build active batch $\mathcal{B}_t$ from hard/anchor pools $\mathcal{H}_t, \mathcal{A}_t$ ▷ Hard/Anchor quota
Optional: refresh `<FewShotExamples>` via synthetic twin-pair generator
Level 1 (evolution)
Run AlphaEvolve on $\mathcal{B}_t$ with active-batch fitness ▷ Hard/Anchor score
Re-evaluate $\mathcal{B}_t$; update $\mathcal{H}_t$ and $\mathcal{A}_t$
Validate on holdout; compute $\mathcal{S}(P)$ ▷ Eq. (5)
Level 2 (consolidation)
Merge top-K archive prompts into P_t^{cons}
if $\mathcal{S}(P_t^{\text{cons}}) \geq \mathcal{S}(P_t) - \delta$ **then**
    $P_{t+1} \leftarrow P_t^{\text{cons}}$
else
    $P_{t+1} \leftarrow$ best Level-1 evolution prompt
end if
if $|\mathcal{H}_t| \leq \theta_{\text{exp}}$ **then**
    Expand $\mathcal{S}_t$ from $\mathcal{U}_t$ (farthest-from-anchor sampling)
end if
return P_{t+1}

Hard examples for diverse error coverage, while Anchor examples preserve already mastered behavior. To avoid overfitting to a narrow cluster of errors, Hard examples are selected via k-means clustering on sentence embeddings (Margatina et al., 2021), taking diverse representatives from each cluster; Anchor examples are drawn with stratified sampling over rating labels. Appendix A reports the embedding model and other run-level hyperparameters.

Pool expansion. As the prompt improves within a cycle, previously hard examples are reclassified as anchors, so $|\mathcal{H}_t|$ typically shrinks even though no examples leave the dataset. When the hard pool falls below a threshold θ_{exp} (e.g., 5–10 examples), the controller *expands* the seen pool by importing new examples from the unseen pool $\mathcal{U}_t$. Candidates are ranked by semantic distance to the current anchor set (cosine distance in embedding space) so that newly added examples are maximally dissimilar from already mastered cases, encouraging coverage of new failure modes rather than redundant near-duplicates.

3.6 Evolutionary Objective

The inner loop follows the AlphaEvolve paradigm: an LLM acts as a mutator in a genetic algorithm, maintaining a diverse archive of prompt candidates (MAP-Elites-style database) and selecting parents for crossover/mutation without gradient access.

Fitness on the active batch. For each candidate prompt P, fitness depends on three active-batch signals: majority-vote accuracy on Hard examples, accuracy on Anchor examples, and mean pairwise Cohen's κ among ensemble members on Hard examples. We subtract a length penalty for prompts longer than $L_{\max}$ tokens to discourage prompt bloat. In our implementation the corresponding weights are $(0.5, 0.3, 0.2)$, prioritizing Hard-batch accuracy while retaining anchors as a regression check (see Section 5.2). The mutator also receives **error artifacts**: misclassified reviews, top confusion pairs (e.g., gold 3 → pred 4), and anchor-regression cases formatted as natural-language feedback.

Combined validation score. For full validation and test evaluation (not the inner-loop fitness), we use a unified combined score that balances global accuracy, worst-group robustness, ordinal error severity, and ensemble agreement:

$$\begin{aligned} C_{\text{val}}(P) = 0.4\, R^{\text{val}}_{\text{global}} + 0.3\, R^{\text{val}}_{\text{worst}} \\ + 0.3 \left(1 - \frac{\text{MAE}^{\text{val}}}{4}\right) \\ + 0.1 \max\left(0, \bar{\kappa}^{\text{val}}\right), \end{aligned} \tag{4}$$

clipped to $[0, 1]$, where $\bar{\kappa}^{\text{val}}$ is the mean quadratic-weighted Cohen's κ between ensemble workers on the validation set. This is *not* the same objective as the inner-loop fitness: C_{val} is computed on the full validation split after each outer cycle, whereas evolutionary fitness is evaluated only on the small active batch.

Outer-cycle selection score. Prompts retained across cycles are ranked on the validation split using a smoothed score that rewards robustness under user shift:

$$\begin{aligned} \mathcal{S}(P) = \overline{C}^{(3)}_{\text{val}}(P) + \lambda_{\text{w}} \cdot \overline{R}^{(3)}_{\text{worst}}(P) \\ - \lambda_{\sigma} \cdot \sigma\big(C^{(3)}_{\text{val}}(P)\big), \end{aligned} \tag{5}$$

where $\bar{\cdot}^{(3)}$ denotes the trailing three-cycle average of per-cycle values, $\sigma(\cdot)$ is the standard deviation over the same window, and we use λ_{w}=0.15,

Method	R_{worst} (↑)	R_{global} (↑)	MAE (↓)	Mean κ (↑)
Fine-tuned baselines (from WILDS leaderboard)				
ERM (Koh et al., 2021)	53.80%	-	-	-
GroupDRO (Sagawa et al., 2020)	53.30%	-	-	-
LISA (Yao et al., 2022)	54.70%	71.30%	-	-
Frozen LLM baselines (Zero-shot)				
Single LLM (`gpt-4o-mini`)	53.18%	71.86%	0.3019	-
Baseline Ensemble (Initial Prompt)	50.00%	71.30%	0.3076	0.6973
Ours (Prompt-only adaptation)				
PRIME	**55.91%**	**75.41%**	**0.2562**	**0.8815**

Table 1: **Main results on the Amazon WILDS full test set.** R_{worst} is the 10th percentile user accuracy (primary robustness metric). PRIME outperforms both the baseline ensemble and state-of-the-art fine-tuning methods (LISA, GroupDRO).

λ_σ=0.30. The best prompt by $\mathcal{S}(P)$ is stored as `best_val_prompt` for final holdout evaluation. Fitness weights (w_1, w_2, w_3)=$(0.5, 0.3, 0.2)$ were chosen to prioritize Hard-batch accuracy, where most evolutionary gains originate; early Home & Kitchen i.i.d. evaluation pilots showed qualitatively similar rankings under moderate perturbations of B, ρ, and (w_1, w_2, w_3), though we did not run an exhaustive grid search (see Section 5.2).

3.7 Consolidation and Full Algorithm

Between outer cycles, we run a **consolidation** step (Level 2). A separate LLM inspects the top-K prompts from the evolutionary archive together with validation metrics and error artifacts. It merges near-duplicate rules, tightens wording, and promotes robust heuristics from `<DynamicRules>` into `<BaseGuidelines>`. A consolidated prompt is accepted only if its validation score does not regress beyond small tolerances compared to the cycle-best evolution prompt; otherwise the evolution winner is kept.

Algorithm 1 summarizes one outer cycle. Figure 1 illustrates how the label report closes the feedback loop between active-batch evaluation and structured-prompt updates. Appendix G provides a more detailed schematic of how cycle-level rule edits are consolidated into the structured prompt over multiple active-learning rounds.

4 Experiments

4.1 Experimental Setup

Dataset. We evaluate our framework on Amazon-WILDS, the Amazon Reviews sentiment dataset from the WILDS benchmark (Koh et al., 2021). The task is 5-way sentiment classification (1 to 5 stars) based on review text. The benchmark contains 539,502 labeled examples across 2,586 reviewer domains, with 5,008 reviewers in the training set and disjoint train/test users. This simulates a realistic deployment scenario where an annotation system must generalize to the idiosyncrasies of unseen reviewers. This is relevant for prompt-based annotation because new users may differ in verbosity, sarcasm, and rating calibration even when the label space remains unchanged. For prompt discovery, we use compact splits constructed for the adaptation run: 375 training-pool reviews, 225 validation reviews, and 225 development-test reviews spanning 21 product categories. After prompt selection, the resulting prompt is evaluated once on the full, uncapped WILDS test set in our local run (34,533 reviews) for the main results in Table 1. Appendix D reports category-level statistics for the prompt-discovery splits.

Ensemble Configuration. To mitigate the high variance inherent to single-LLM zero-shot classification, we employ a heterogeneous ensemble of three frozen, API-accessible models: `gpt-4o-mini` (OpenAI, 2024), `gemini-2.5-flash` (Google Cloud, 2025), and `claude-3.5-haiku` (Anthropic, 2024). We intentionally combine model families because data labeling benefits when annotators make partially independent errors: majority aggregation reduces noise in human annotation (Snow et al., 2008), and LLM ensembling improves outputs when model strengths are complementary (Jiang et al., 2023; Du et al., 2024). The ensemble's final prediction is majority vote, with all models receiving the same evolved prompt. The mutator LLM driving AlphaEvolve is `gemini-2.5-pro` (Google Cloud,

2025). The main prompt-discovery configuration used to obtain the final prompt is summarized in Appendix A.

Fine-tuned baselines. ERM is the standard WILDS DistilBERT-base-uncased classifier (Sanh et al., 2019), trained with cross-entropy and selected by OOD validation performance. GroupDRO uses the same backbone and training hyperparameters, but optimizes a group-robust objective over reviewer/domain groups to reduce worst-group loss. LISA is a DistilBERT-based selective manifold-mixup method that interpolates same-label examples across domains to learn invariant predictors.

Metrics. Following the standard WILDS protocol, our primary metric is worst-group accuracy (R_{worst}), defined as the 10th percentile of accuracy across all users in the test set. This metric heavily penalizes models that overfit to majority sub-populations while failing on minority groups. We also report global accuracy (R_{global}) over all test examples, Mean Absolute Error (MAE) to capture the severity of ordinal misclassifications, and mean pairwise Cohen's κ to measure inter-annotator agreement within the ensemble.

4.2 Main Results

Table 1 presents the performance of PRIME against both fine-tuning baselines and frozen-LLM approaches on the full, uncapped WILDS test set.

Comparison with LLM baselines. The baseline ensemble, equipped with a standard zero-shot instruction, achieves an R_{worst} of only 50.00%, underperforming even a single `gpt-4o-mini` model (53.18%). This highlights a known failure mode of ensembling: if the prompt is ambiguous, heterogeneous models will fail in diverse ways, destroying the consensus required for a reliable majority vote (evidenced by a low κ of 0.6973). PRIME, by evolving the prompt to clarify these ambiguities, boosts R_{worst} by an absolute 5.91% and raises ensemble agreement κ to 0.8815. Furthermore, the MAE drops significantly from 0.3076 to 0.2562, indicating that the evolved prompt effectively eliminates severe misclassifications (e.g., confusing 1 star with 5 stars).

Comparison with fine-tuning methods. Remarkably, our prompt-only adaptation outperforms LISA (54.70%), a state-of-the-art method that requires computationally expensive fine-tuning of the model's weights. Appendix C summarizes the public ERM/GroupDRO/LISA baseline configurations used for this comparison. This demonstrates that for complex reasoning tasks like sentiment annotation, refining the task definition (the prompt) can be a more effective domain adaptation strategy than updating the model's internal representations.

5 Analysis and Ablation

Unless noted otherwise, ablations use the same ensemble and evaluation protocol as Section 4, with possible differences in train-pool size or optional modules.

5.1 Evolution Dynamics and the Impact of Active Learning

We compare full PRIME (active learning with synthetic few-shot enabled) against a *Plain Evolution* baseline during prompt discovery. In Plain Evolution, AlphaEvolve optimizes the prompt on a fixed, randomly sampled subset of the training pool (matched in size to our active batch), without Hard/Anchor partitioning, pool expansion, or synthetic few-shot injection.

Figure 2 plots four metrics on the representative development test split per outer AL cycle (Base, cycles 0–5, and Final). Plain Evolution improves only marginally after the first cycle: R_{global} rises from $\sim$0.76 to $\sim$0.78 and then flattens, while R_{worst} jumps once at cycle 1 ($\sim$0.53 $\rightarrow$ $\sim$0.59) and remains nearly flat thereafter. PRIME reaches substantially higher ceilings on every metric. For example, R_{worst} peaks at $\sim$0.67 (vs. $\sim$0.59 for Plain Evolution) and MAE drops to $\sim$0.18 at cycle 5 (vs. $\sim$0.23–0.24).

PRIME is not monotonic: all four panels show a synchronized regression at cycle 4 (e.g., R_{worst} falls from $\sim$0.67 to $\sim$0.56), consistent with a difficult pool-expansion or batch-refresh event, followed by recovery at cycle 5 and a strong Final checkpoint.

The combined score, computed with the same weighting as Eq. (4), mirrors this pattern and peaks at $\sim$0.91 for PRIME versus $\sim$0.865 for Plain Evolution. Overall, active batch rotation drives higher variance but unlocks a higher robustness ceiling than static-batch evolution.

5.2 Hyperparameter Sensitivity

Our main all-categories configuration fixes B=80, ρ=0.7, fitness weights $(0.5, 0.3, 0.2)$, and selection weights $(\lambda_{\text{w}}, \lambda_{\sigma})$=$(0.15, 0.30)$ after pilot runs

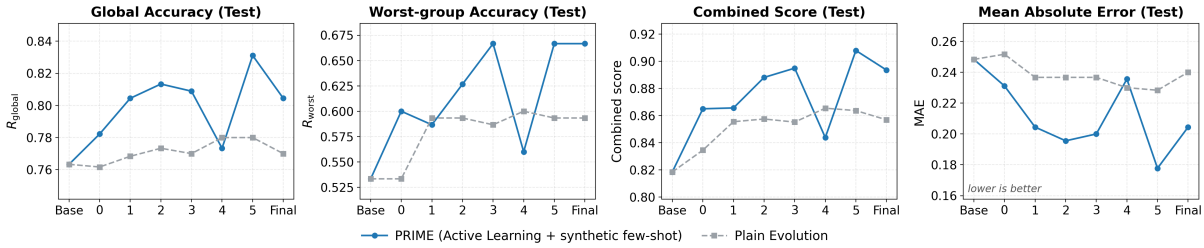

Figure 2: **Metric evolution per AL cycle (development test split).** Four metrics tracked from the Base prompt through outer cycles 0–5 and the Final checkpoint during prompt discovery. **Blue (solid):** PRIME with active learning and synthetic few-shot. **Grey (dashed):** Plain Evolution on a fixed training subset. PRIME reaches higher peak and final values on R_{global}, R_{worst}, and combined score, and achieves lower MAE. Plain Evolution remains comparatively flat after its early gains, while PRIME shows a temporary regression at cycle 4 followed by strong recovery.

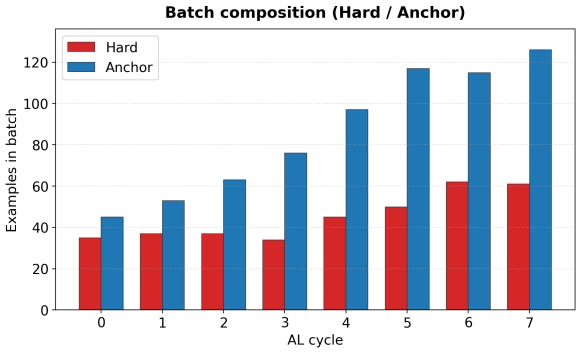

Figure 3: **Active Pool Dynamics.** As the prompt evolves, examples migrate from the Hard pool to the Anchor pool. Periodic spikes indicate pool expansion events where new challenging examples are imported from the Unseen pool.

on a Home & Kitchen subsample. We did not perform an exhaustive grid search over all knobs. Informal checks varying $B \in \{60, 80, 100\}$ and $\rho \in \{0.6, 0.7, 0.8\}$ preserved the qualitative ranking (PRIME $>$ plain evolution) and similar validation curves, which motivated locking the default values before the final full-test evaluation. Because evolutionary search is stochastic, exact scores can shift by roughly 0.5–1.0 percentage points on R_{worst} under API noise; we therefore report full-test numbers from a single locked configuration rather than tuning hyperparameters on the test set.

Finally, Appendix E reports the synthetic few-shot ablation. Removing the generator reduces Hard-pool accuracy from 38.0% to 26.8%, supporting its role in clarifying adjacent rating boundaries.

5.3 Active Pool Dynamics

The success of PRIME relies on the continuous migration of examples from the Hard pool to the Anchor pool.

Figure 3 illustrates how examples migrate from Hard to Anchor across active-learning cycles. As prompts resolve current failures, Hard examples move to Anchor; when Hard is depleted below θ_{exp}, pool expansion imports semantically distant unseen examples, producing controlled spike-and-recovery events. These spikes indicate deliberate exposure to new data regions rather than optimization failure, while recovery shows that PRIME is improving coverage rather than memorizing one static batch.

6 Conclusion

In this work, we introduced PRIME, a framework that adapts an ensemble of large language models to a target annotation domain entirely through prompt optimization. By coupling AlphaEvolve-style mutation with dynamic Hard/Anchor data pools, PRIME keeps prompt search focused on unresolved edge cases, reducing premature convergence and catastrophic forgetting in prompt space.

On Amazon WILDS, PRIME improves upon the zero-shot baseline ensemble, increasing worst-group accuracy (R_{worst}) by nearly 6% absolute, and outperforms state-of-the-art fine-tuning methods such as LISA. Ablation studies confirmed that both the active-learning loop and the injection of synthetic boundary examples are critical for refining the prompt's decision boundaries and resolving the hardest edge cases. These gains suggest that much of the remaining annotation error comes from underspecified task instructions rather than from missing model capacity alone. Overall, rigorous, data-driven instruction tuning provides a scalable alternative to traditional fine-tuning for complex annotation tasks, especially when models are accessed only through inference APIs.

Limitations

While PRIME provides a highly effective prompt-only mechanism for domain adaptation, it introduces specific trade-offs that practitioners should consider.

API Cost vs. Compute Cost. PRIME shifts the burden of domain adaptation from GPU compute (required by fine-tuning methods such as LISA or LoRA) to API token usage. The evolutionary search, combined with active batch evaluation and consolidation, requires tens of thousands of API calls to the mutator and ensemble models. Although our active-learning controller significantly reduces this cost compared to evaluating on the full training set, the prompt discovery phase remains more expensive in terms of API billing than a simple zero-shot approach. Appendix B provides a token-level cost estimate for the main run.

Inference Token Overhead. The evolved structured prompt, which includes detailed `<DynamicRules>` and `<FewShotExamples>`, is inherently longer than a monolithic zero-shot instruction. Despite our fitness penalty for prompt length ($\mathcal{P}_{\text{len}}$), applying the final evolved prompt to millions of unlabeled examples in a production setting will incur higher per-example inference costs due to the increased input token count.

Dependence on Pretrained Knowledge. Because the ensemble remains frozen, PRIME's success is bounded by the latent capabilities of the underlying LLMs. For tasks requiring highly specialized, proprietary, or rapidly changing factual knowledge not present in the models' pretraining data, prompt optimization alone cannot inject new facts. In such scenarios, traditional fine-tuning or Retrieval-Augmented Generation (RAG) remains necessary.

References

Anthropic. 2024. Model card addendum: Claude 3.5 Haiku and upgraded Claude 3.5 Sonnet. https://assets.anthropic.com/m/1cd9d098ac3e6467/original/Claude-3-Model-Card-October-Addendum.pdf. Accessed 2026-05-26.

Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. 2024. Active prompting with chain-of-thought for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1330–1350. ArXiv:2302.12246.

Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Boyang Li, Shafiq Joty, and Lidong Bing. 2023. Is GPT-3 a good data annotator? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 11173–11195. ArXiv:2212.10450.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 11733–11763. ArXiv:2305.14325.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active learning for BERT: An empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. ArXiv:2309.16797.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences (PNAS)*, 120(30):e2305016120.

Google Cloud. 2025. Gemini 2.5 Flash and Gemini 2.5 Pro. https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash. Model documentation. Accessed 2026-05-26.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations (ICLR)*. ArXiv:2309.08532.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8342–8360. ArXiv:2004.10964.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*. ArXiv:2106.09685.

Badr Youbi Idrissi, Martín Arjovsky, Mohammad Pezeshki, and David Lopez-Paz. 2022. Simple data balancing achieves competitive worst-group accuracy. In *Conference on Causal Learning and Reasoning (CLeaR)*, pages 336–351. ArXiv:2110.14503.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 14165–14178. ArXiv:2306.02561.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations (ICLR)*. ArXiv:2310.03714.

Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. 2024. Analyzing dataset annotation quality management in the wild. *Computational Linguistics*, 50(3):817–866.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, and 5 others. 2021. WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 5637–5664. ArXiv:2012.07421.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations (ICLR)*. ArXiv:2202.10054.

Evan Z. Liu, Behzad Haghgoo, Annie S. Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. 2021. Just train twice: Improving group robustness without training group information. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 6781–6792. ArXiv:2107.09044.

Katerina Margatina, Timo Schick, Nikolaos Aletras, and Jane Dwivedi-Yu. 2023. Active learning principles for in-context learning with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5011–5034. ArXiv:2305.14264.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active learning by acquiring contrastive examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 650–663. ArXiv:2109.03764.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics (TACL)*, 12:933–949. ArXiv:2401.00595.

OpenAI. 2024. GPT-4o system card. `https://cdn.openai.com/gpt-4o-system-card.pdf`. Accessed 2026-05-26.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7957–7968. ArXiv:2305.03495.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475.

Marta Sabou, Kalina Bontcheva, Leon Derczynski, and Arno Scharl. 2014. Corpus annotation through crowdsourcing: Towards best practice guidelines. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC)*, pages 859–866.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2020. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations (ICLR)*. ArXiv:1911.08731.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*. ArXiv:1910.01108.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations (ICLR)*. ArXiv:2310.11324.

Burr Settles. 2009. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison.

Asankhaya Sharma. 2025. OpenEvolve: an open-source evolutionary coding agent.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ArXiv:2402.13446.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205. ArXiv:2108.13487.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*. ArXiv:2309.03409.

Huaxiu Yao, Yu Wang, Sai Li, Linjun Zhang, Weixin Liang, James Zou, and Chelsea Finn. 2022. Improving out-of-distribution robustness via selective augmentation. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*. ArXiv:2201.00299.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*. ArXiv:2211.01910.

A Main Experiment Configuration

Table 2 summarizes the configuration of the main prompt-discovery run used to obtain the final PRIME prompt reported in Table 1. We omit implementation-only settings such as logging, checkpoint paths, retry counts, and the full mutator instruction text; the table retains the parameters that define the experimental protocol.

Component	Setting
Outer AL cycles	8
Evolution steps per cycle	15
Evolution implementation	OpenEvolve implementation of AlphaEvolve (Sharma, 2025)
Mutator model	`gemini-2.5-pro` (Google Cloud, 2025), temperature 0.6
Worker ensemble	`gpt-4o-mini` (OpenAI, 2024); `gemini-2.5-flash` (Google Cloud, 2025); `claude-3.5-haiku` (Anthropic, 2024)
Worker decoding	Temperature 0.0; majority vote over 3 workers
Active batch	B=80, Hard ratio ρ=0.7
Hard threshold	τ=0.0 for normalized worker disagreement
Embedding model	`all-MiniLM-L6-v2` for clustering and farthest-from-anchor sampling
Fitness	$0.5\,\mathrm{Acc}_{\mathrm{Hard}} + 0.3\,\mathrm{Acc}_{\mathrm{Anchor}} + 0.2\,\kappa_{\mathrm{Hard}} - \mathcal{P}_{\mathrm{len}}$
Synthetic few-shot	Enabled; `inject_as_hint`; 8 examples; 4 boundary pairs
Pool expansion	Triggered below 15 Hard examples; refresh 10 examples per cycle
Evolution archive	Population 30; archive 500; 5 feature bins
Final prompt length	7148 tokens
Dataset	WILDS Amazon Reviews, all categories, user-disjoint splits

Table 2: Configuration of the main prompt-discovery run. The compact train/validation/development-test splits drive evolution and prompt selection; Table 1 reports the selected prompt on the full WILDS test set.

B API Usage and Cost Estimate

Table 3 summarizes token usage for the main prompt-discovery run. The reported logs account for ensemble evaluation and consolidation calls; mutator-token accounting was not available in the same log, so we estimate it as 8 cycles $\times$ 15 evolution steps, with approximately 30,000 input and 3,000 output tokens per mutation call. Costs assume no prompt caching, batch discount, or free-tier usage, using per-million-token prices of `gemini-2.5-flash` ($0.30 input, $2.50 output), `gpt-4o-mini` ($0.15 input, $0.60 output), `claude-3.5-haiku` ($0.80 input, $4.00 output), and `gemini-2.5-pro` ($1.25 input, $10.00 output).

Call source	Input tokens	Output tokens	Cost
`gemini-2.5-flash` workers	12.15M	0.03M	$3.71
`gpt-4o-mini` workers	11.51M	0.04M	$1.75
`claude-3.5-haiku` workers	13.15M	0.51M	$12.54
Consolidation, `gemini-2.5-pro`	0.07M	0.05M	$0.58
Mutator estimate, `gemini-2.5-pro`	3.60M	0.36M	$8.10
Total (reported + estimated)	**40.48M**	**0.98M**	**$26.67**

Table 3: Approximate API cost of the main run. Reported calls account for 37.50M tokens and approximately $18.57; adding the mutator estimate yields approximately 41.46M tokens and $26.67.

C Fine-Tuned Baseline Details

Table 4 summarizes the public fine-tuned baselines used in Table 1. We do not rerun these baselines; we report the published Amazon-WILDS results from WILDS and LISA under the standard user-disjoint evaluation protocol.

D Dataset Category Distribution

Table 5 shows the distribution of reviews across product categories for the prompt-discovery splits summarized in Table 2. The training pool is sampled via clustering to ensure diverse coverage, while the validation and development test splits are constructed representatively across categories. The final prompt selected from this process is then evaluated on the full WILDS test set in Table 1.

E Synthetic Few-Shot Ablation

In the main WILDS experiments (Table 1), synthetic few-shot generation is enabled. To isolate its impact, we compare the full PRIME pipeline against an otherwise identical variant without synthetic few-shot generation. The generator targets adjacent-class confusions by creating minimal review pairs that differ in one rating-relevant cue. Injecting these boundary examples into `<FewShotExamples>` improves Hard-pool accuracy from 26.8% to 38.0%, indicating better calibration around difficult rating boundaries.

Baseline	Critical training hyperparameters (source)
ERM	DistilBERT-base-uncased (Sanh et al., 2019); cross-entropy ERM; batch size 8; learning rate 1×10^{-5} with AdamW; L2 regularization 0.01; max sequence length 512; 3 epochs with early stopping on OOD validation; learning rate selected from $\{1\times10^{-6}, 2\times10^{-6}, 1\times10^{-5}, 2\times10^{-5}\}$ (WILDS (Koh et al., 2021))
GroupDRO	Same DistilBERT-base-uncased (Sanh et al., 2019) backbone and ERM hyperparameters; group-robust objective over reviewer/domain groups; GroupDRO step size fixed to 0.01 (WILDS (Koh et al., 2021; Sagawa et al., 2020))
LISA	DistilBERT backbone; selective manifold mixup for Amazon; intra-label strategy only (p_{sel}=1.0); $\lambda \sim \text{Beta}(2, 2)$; learning rate 2×10^{-6}; batch size 8; Adam optimizer; weight decay 0; maximum 3 epochs (LISA (Yao et al., 2022))

Table 4: Critical configuration details for fine-tuned baselines compared against PRIME. All baselines are trained on labeled WILDS data and evaluated with the 10th-percentile user accuracy protocol used in Table 1.

ID	Category Name	Train	Validation	Dev. Test
0	Books	196	110	97
1	Kindle_Store	24	26	21
2	Video_Games	1	0	1
4	Arts_Crafts_and_Sewing	1	0	2
5	Toys_and_Games	14	0	2
6	Automotive	4	0	2
7	Tools_and_Home_Improvement	16	1	7
8	Sports_and_Outdoors	4	3	3
10	Prime_Pantry	5	0	1
11	Pet_Supplies	9	12	7
12	CDs_and_Vinyl	10	7	15
13	Patio_Lawn_and_Garden	3	0	3
14	Office_Products	3	2	8
15	Cell_Phones_and_Accessories	2	0	2
16	Clothing_Shoes_and_Jewelry	12	5	11
18	Movies_and_TV	15	26	6
20	Electronics	13	8	7
21	Grocery_and_Gourmet_Food	19	10	11
23	Luxury_Beauty	0	0	1
24	Home_and_Kitchen	23	14	18
25	Industrial_and_Scientific	1	1	0

Table 5: Distribution of reviews across product categories in the prompt-discovery splits.

F Synthetic Few-Shot Example Set

Figure 4 shows an illustrative set of synthetic few-shot examples generated for the Amazon-Reviews task. The generator produces only the review texts and target ratings; the highlights in the figure are added manually for visualization. The examples cover the full ordinal scale and illustrate the cues that should move a review across adjacent rating boundaries.

G Consolidation and Rule Promotion Detail

Figure 5 illustrates the interaction between active-learning cycles and consolidation. The left side shows cycle-level rule edits, including additions and removals for rating boundaries and edge cases. After each cycle, consolidation filters these edits, promoting stable rules into `<BaseGuidelines>` while retaining more tentative, boundary-specific heuristics in `<DynamicRules>`.

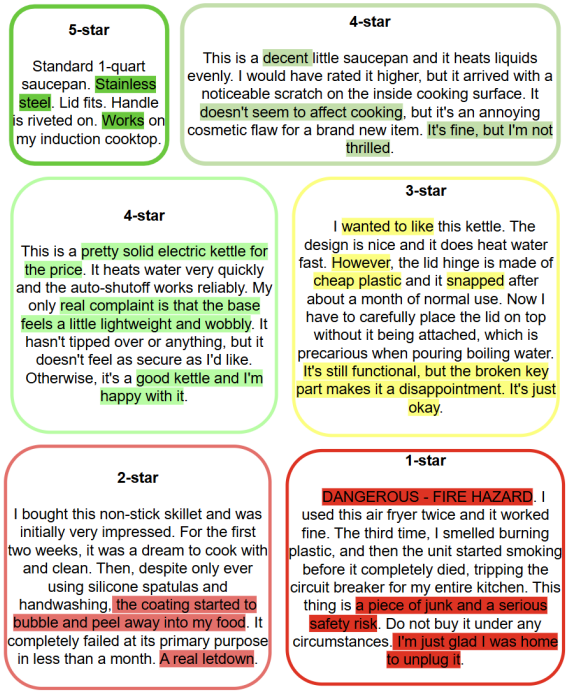

Figure 4: Synthetic few-shot examples for ordinal sentiment boundaries. The generator creates controlled review texts covering positive, mixed, negative, and severe-failure cases. Highlights are manual visual annotations, not generator output. The generated texts are injected into `<FewShotExamples>` to clarify adjacent class boundaries for the LLM ensemble.

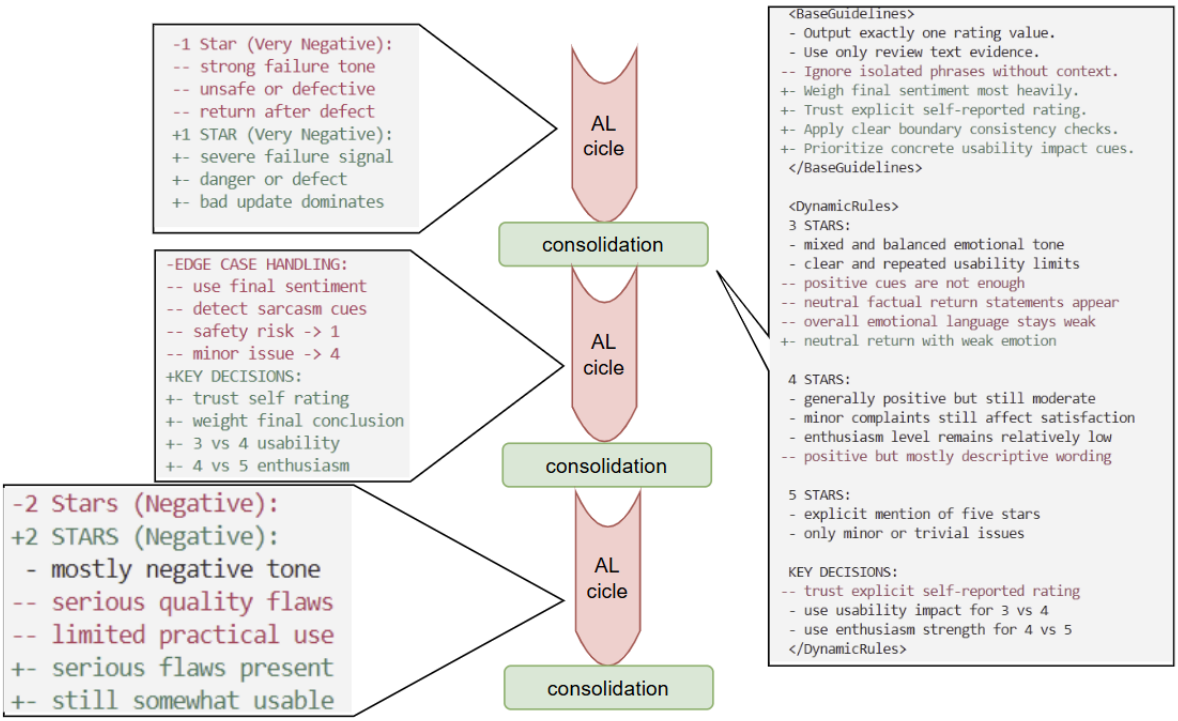

Figure 5: Detailed consolidation process across active-learning cycles. Cycle-level edits are repeatedly produced during active learning, then consolidated into the structured prompt. Stable, broadly useful rules are promoted into `<BaseGuidelines>`, while more local rating-boundary heuristics remain in `<DynamicRules>`.