Bad Joke Project: Prompt Discovery to Re-Inject "Friction" into 0–10 Ratings

Finno

Overview and Motivation

This project is a toy but concrete mitigation demo: can we make a state-of-the-art chat model give more *honest, discriminative* feedback (including negative feedback) instead of defaulting to the socially convenient move of being nice? In the broader course framing, the aim was to "re-inject friction"—i.e., restore the kind of mild social resistance that helps calibration and learning, rather than unconditional affirmation.

I operationalized this with a deliberately low-stakes domain: rating jokes on a 0–10 scale. Jokes are useful here because they are (1) naturally subjective, (2) strongly shaped by social politeness pressures, and (3) easy to score with a single integer. The core hypothesis was that a carefully discovered *system prompt* could reduce sycophantic inflation and improve alignment with human ratings, without changing model weights.

Data Collection and Cleaning

I used two public sources: a Kaggle jokes/ratings dataset (Jester-style humor ratings) and an r/Jokes Reddit dataset. My first step was purely mechanical: parse both sources into a consistent, "workable" tabular format with joke text and a target rating in $(0,1,\ldots,10)$. For the Kaggle set, I rescaled/centered ratings into the 0–10 range and experimented with per-user mean normalization (subtracting each rater's mean to reduce individual generosity/harshness bias). In hindsight, this normalization may have been unnecessary and could even distort the target by removing meaningful variance.

To make evaluation interpretable, I created balanced splits. The Reddit-derived files were constructed so each discrete rating bucket was equally represented: 20 examples per rating (yielding $11 \times 20 = 220$) for training and 10 per rating (yielding $11 \times 10 = 110$) for validation. I also produced a small, centered Jester subset for optional training augmentation.

Method: Black-Box Prompt Search via Loss Feedback

The system has two roles: an **evaluator** and a **proposer**. The evaluator calls the OpenAI API, asks the model to rate a provided joke from 0 to 10, parses the returned integer, and compares it to the human target. I tracked mean absolute error (MAE) and overall bias (mean prediction minus mean target). Crucially, I also computed *bucketed mean signed error* by target rating (e.g., are "8"-rated jokes being inflated by +2.5 on average?). This bucket signal became the pseudo-gradient for improvement.

The proposer is a second API call whose job is to update the *system prompt* given the previous prompt(s) and their error profile. The proposer is explicitly instructed to write a domain-general

'Evaluation Mode" paragraph that avoids sycophancy, avoids collapsing everything to the middle, and avoids overcorrecting into harshness. To keep the discovered instruction general (not 'joke-specific"), I added lexical constraints forbidding dataset/domain words (e.g., 'joke", 'reddit", "jester") and fell back to a safe generic seed prompt if violated.

After iterating, I ran an "exam" sweep of 50 iterations (stopping largely due to cost; the run consumed millions of tokens). Each iteration produced a prompt plus train/validation metrics and bucket diagnostics logged to disk.

Interactive Demo

Finally, I deployed a Hugging Face/Gradio app that lets a user pick an iteration id (e.g., 1–50) and compare side-by-side outputs: a baseline friendly assistant system message versus the selected discovered evaluation prompt. The UI also reveals the exact system prompt text and the iteration's metrics, making the tradeoff between 'pleasant" and 'calibrated" visible rather than theoretical.

Limitations and Next Steps

This is not gradient descent; it is a noisy black-box search where the proposer itself can introduce variance and prompt overfitting. The dataset is small and subjective, and per-user normalization may have removed legitimate structure. Next steps would include: larger human-labeled datasets across domains, stronger holdout tests for generalization, and explicit measurement of "range use" (does the model actually use 0–10 meaningfully rather than clustering?). Still, as a toy demo, the pipeline shows a key point: prompt tuning alone can measurably shift a model away from reflexive agreeableness and toward more human-aligned scoring.

Chart Analysis

Figure 1 shows mean absolute error (MAE) across prompt-discovery iterations. The first ≈ 25 runs are noticeably noisier because each evaluation used a small sample (often ~ 20 jokes), so the measured loss had high variance. From runs 25–50, the curves stabilize and suggest that prompt updates can modestly improve average error, but improvements are not monotonic and remain sensitive to the particular validation sample.

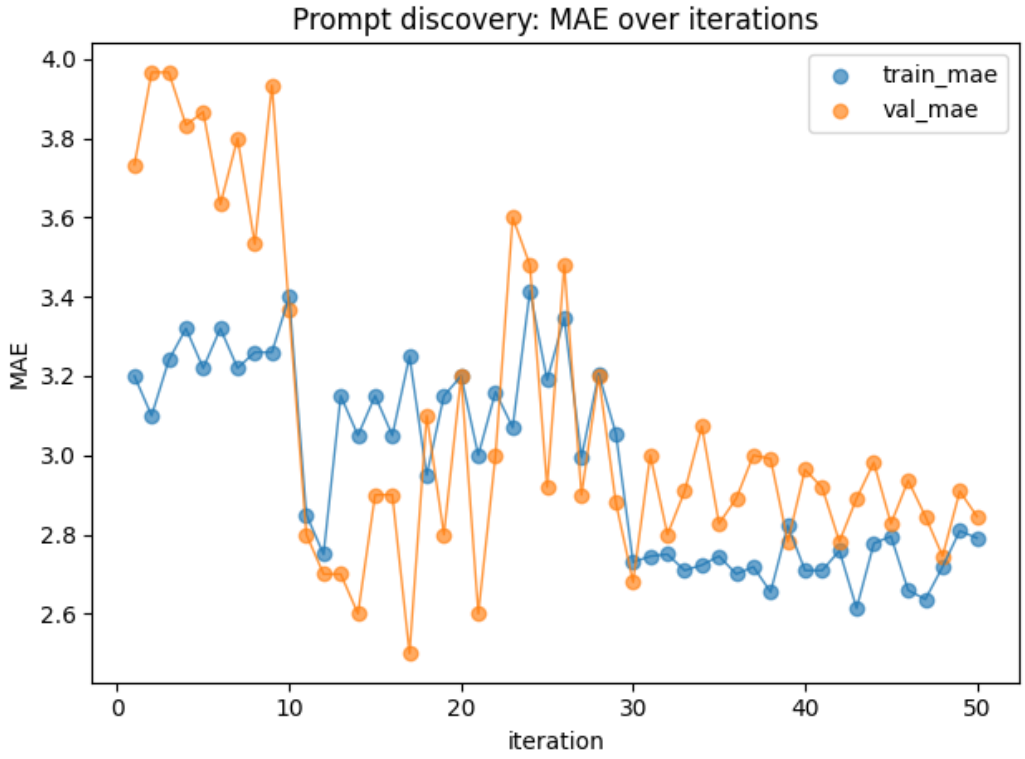

Figure 1: MAE over prompt-discovery iterations (train vs. validation). Early runs fluctuate due to small evaluation sample size.

Figure 2 compares bucketed signed error (prediction minus target) for two checkpoints (run 25 vs. run 50). The model exhibits the classic "play it safe" pattern: it over-predicts low-rated jokes (positive error for buckets 0–3) and under-predicts high-rated jokes (negative error for buckets 7–10). Notably, this extremal bias does not disappear—and for the endpoints (0 and 10) it can even worsen—despite the later prompts explicitly instructing the model to "use the full range" rather than softening toward the middle. This supports the broader lesson of the project: system prompting can steer behavior, but it does not reliably overwrite entrenched calibration habits learned during pretraining.

Figure 3 visualizes the final run's rating profile on a validation set with an even true-label distribution (10 examples per rating). The model's predicted counts are highly concentrated in the mid-range, especially 3–7, while it almost never uses the extremes. Interestingly, it also under-uses the literal midpoint (5). A plausible interpretation is that the prompt's instruction to avoid "defaulting to the middle" was internalized as "avoid 5" rather than "avoid central clustering," which is a good example of how prompt constraints can be misread in a literal, local way.

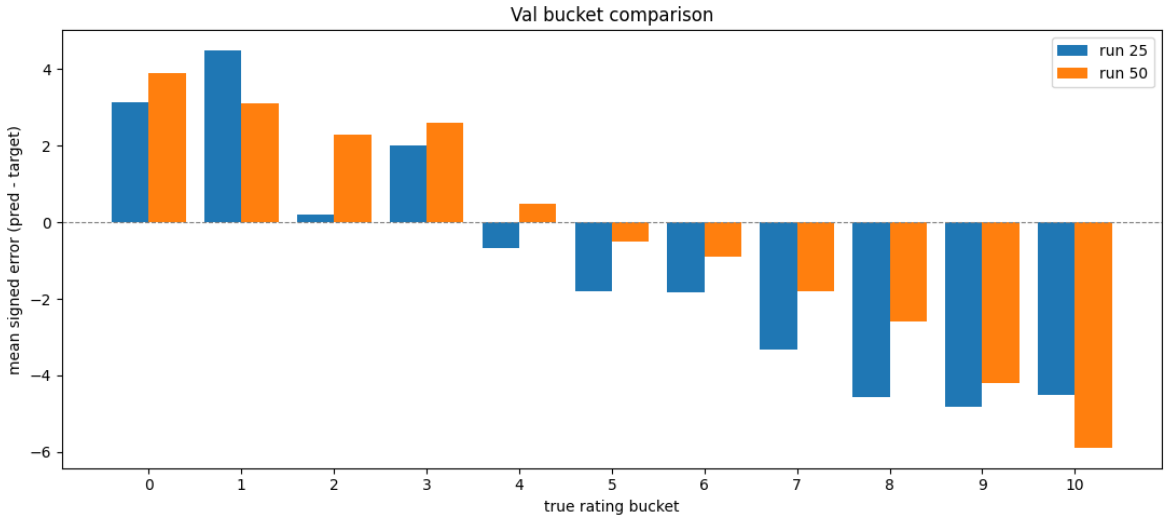

Figure 2: Validation bucket comparison: mean signed error (pred – target) by true rating bucket, contrasting run 25 and run 50.

Finally, Figures 4–5 plot bucketed signed error trajectories from runs 25–50. The curves show that some buckets partially drift toward reduced bias, but the overall shape persists: low buckets remain positively biased and high buckets remain negatively biased. The "windowed" view makes this comparability clearer across buckets, highlighting that the prompt updates tend to yield small, incremental shifts rather than a regime change in calibration.

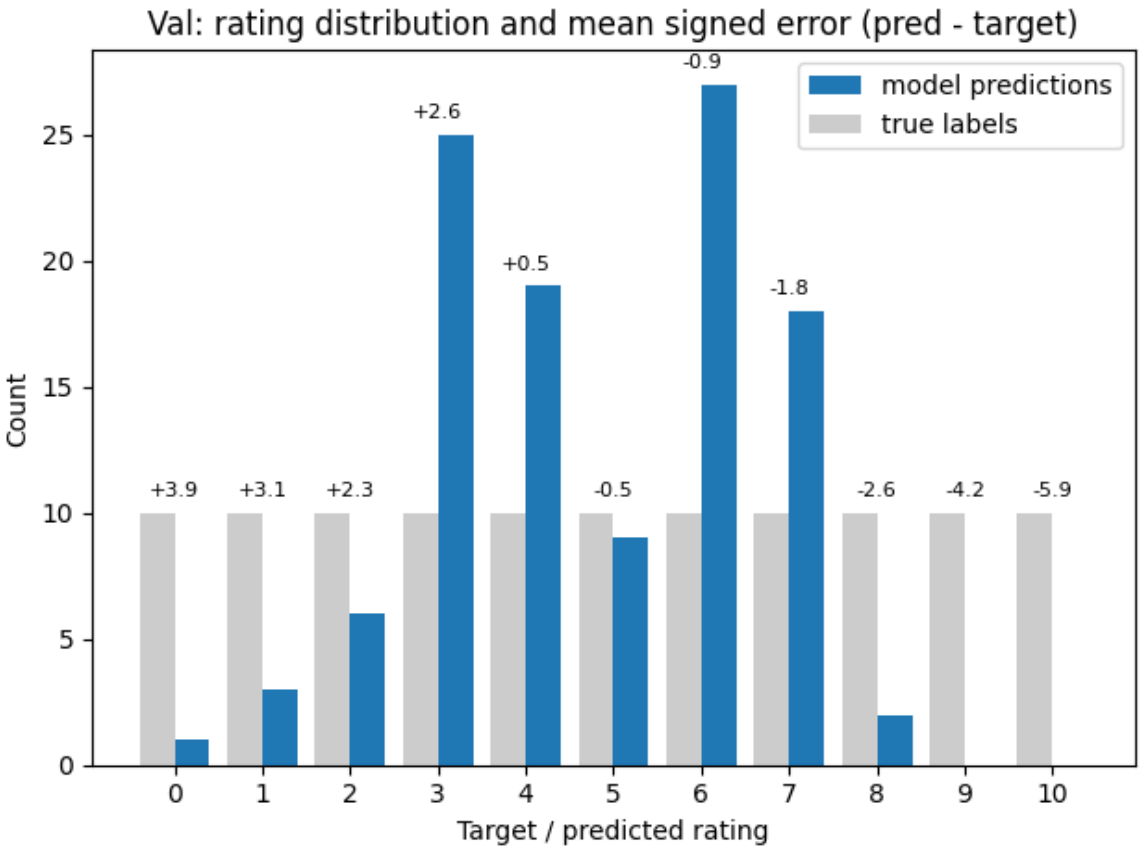

Figure 3: Run 50 validation: predicted rating counts versus true (uniform) labels, with mean signed error by bucket annotated.

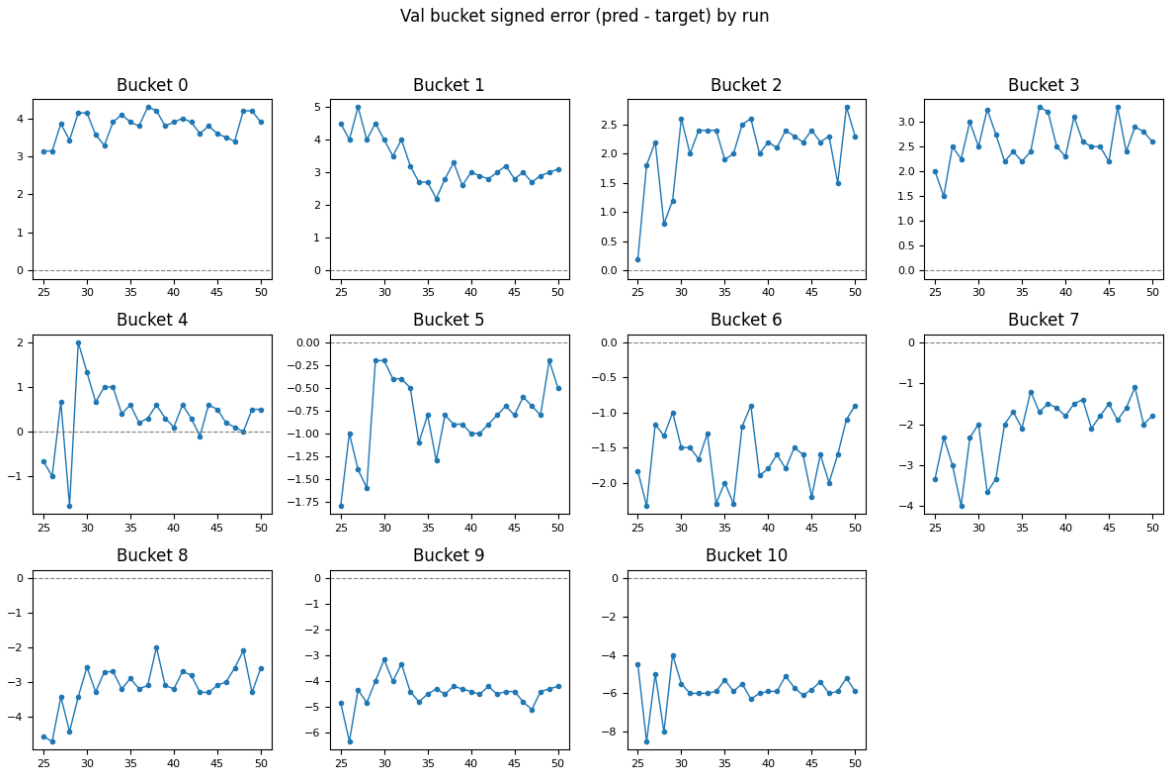

Figure 4: Bucketed signed error trends (runs 25–50).

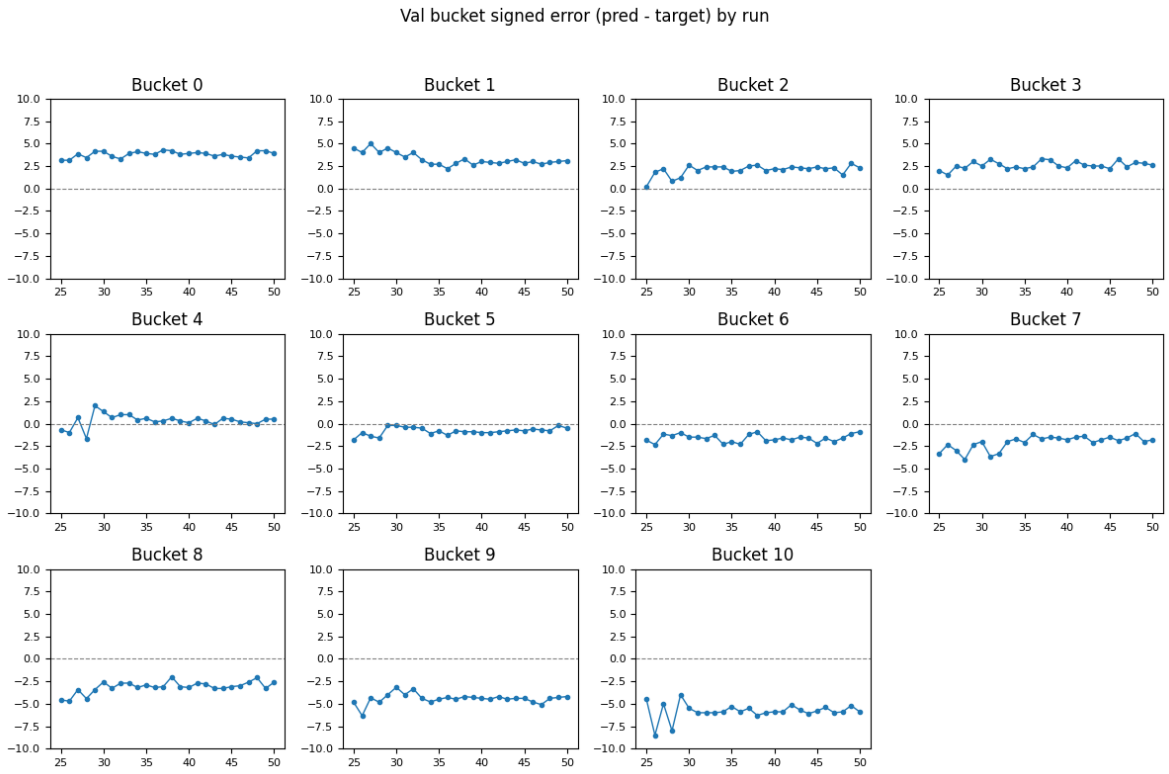

Figure 5: Bucketed signed error trends with a constant y-axis window for easier bucket-to-bucket comparison.