Prompt Tuning for Growth Opportunities using Open Source Collaboration Data

1. Summary

Following our initial finding that Open Source data is structurally too sparse (1.1% network density) to trigger Enterprise-grade "Pattern Recognition," we conducted two experimental tuning runs to identify if the engine could be adapted to this environment.

The main finding was that we successfully forced the engine to generate insights (moving from 0 → 192 Growth Opportunities). However, the data reveals a clear trade-off: **Test A** maintained high precision but still a very low volume, while **Test B** generated high volume but shifted towards "transactional" feedback (e.g., fixing typos) rather than developmental coaching.

2. Introduction

2.1 Context

The previous test (pandas-initial) confirmed that the standard Enterprise-aimed model/prompts failed to generate Growth Opportunities on Open Source data, even though the data was filtered to capture active users. The engine's default logic requires "recurring behavioral patterns" across multiple reviewers, which from the first test was never met.

2.2 Objective

The goal of this phase was to test if **Prompt Engineering** alone solve this issue, and to find a sweet spot for getting growth opportunities. How lenient or strict does the prompt have to be to:

1. **Recall:** Get enough data to be useful.
2. **Precision:** Keeping the advice strategic and developmental.

3. Methodology

Including the last test (baseline), so far we have compared three prompt configurations on the same dataset (Pandas, Consistent Contributors).

3.1 Baseline (Previous State)

- **Logic:** Standard Enterprise Model. Required "recurring behavioral patterns."
- **Outcome: 0** Growth Opportunities.

3.2 Test Run A: "Individual Signal" (Seed 4)

- **The Change:** We modified the System Prompt to explicitly allow **a single high-quality piece of constructive feedback** to trigger a Growth Opportunity, removing the requirement for repetition. Overall the prompt stayed very similar, only changing a few lines.
- **Hypothesis:** This should capture the "drive-by" coaching style common in Open Source.

3.3 Test Run B: "Radical Simplification" (Seed 5)

- **The Change:** We removed specific role constraints and broadened the definition of "Growth" to include any actionable improvement, regardless of its developmental depth.
- **Hypothesis:** This was a "Stress Test" to find the ceiling of the engine's recall capability.

```
Rules
- Coach, not critic: respectful, practical, and grounded in evidence.
- Focus on observable actions in the work; avoid personality, motives, or
speculation.
- Use ONLY reviewer-authored comments as evidence. Recipient comments are
background only.
- Inputs are ordered newest → oldest by updatedAt.
- If there is no clear constructive signal, return [].

Task
- Identify up to 4 meaningful growth themes implied by reviewer comments.
- Prefer patterns across multiple comments; avoid duplicates and one-off
nitpicks.

Output requirements (each item)
- title: action-oriented, starts with a verb, ≤ 50 characters. Reuse an
existing matching title if available.
- commentIds: reviewer commentIds that support the theme (reviewer-only).
```

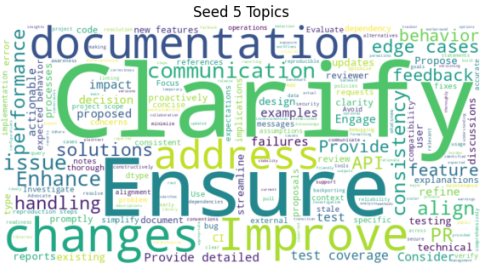
4. Results & Comparative Analysis

4.1 Quantitative Output

- **Baseline:** 0 Growth Opportunities.
- **Test A (Seed 4): 5** Growth Opportunities. (High Precision).
- **Test B (Seed 5): 192** Growth Opportunities. (High Recall).

4.2 Topic Analysis

To understand the quality of the advice being generated, we compared the topics extracted by both models.

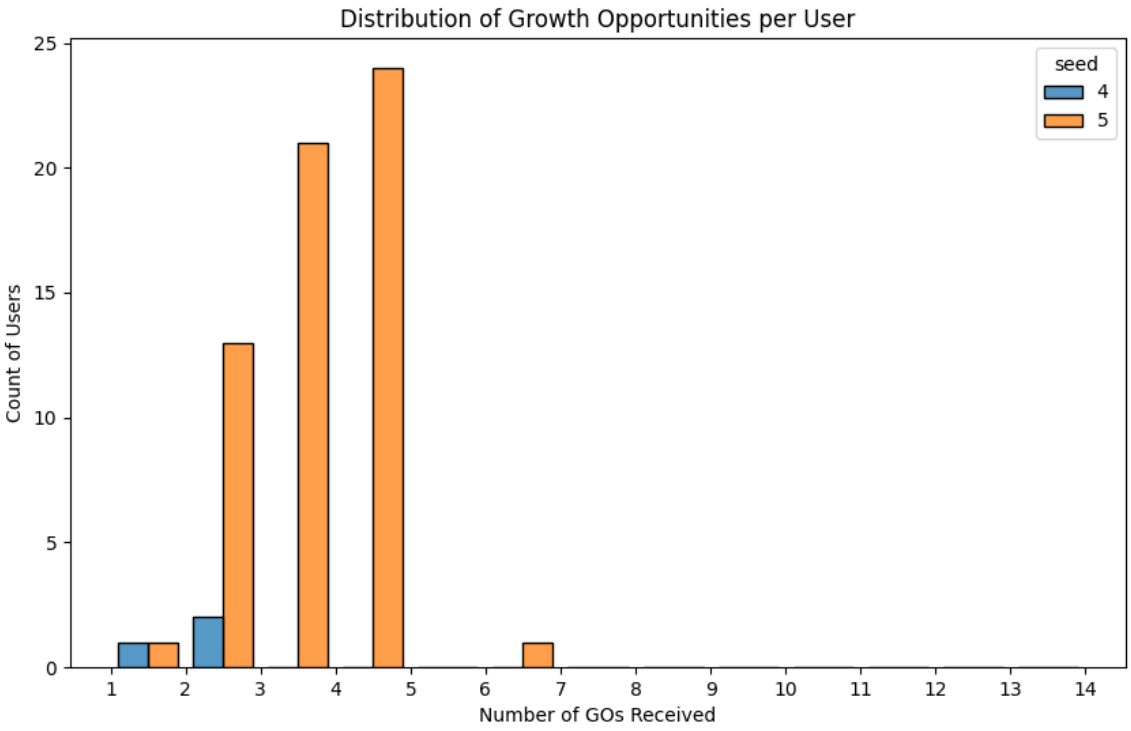


Observation:

- **Seed 4 (Left):** The topics remain strategic and behavioral (e.g., *"Proactively," "Communication"*). This suggests the "Individual Signal" prompt still filters for quality.
- **Seed 5 (Right):** The word cloud shifts significantly towards transactional verbs (e.g., *"Fix," "Update," "Check"*). This indicates that the "Radical" prompt began interpreting code-level corrections (nitpicks) as personal growth opportunities.

4.3 "Spaminess" & Distribution

A key risk of lowering thresholds is overwhelming the user. We analyzed how many Growth Opportunities were assigned to each unique user.

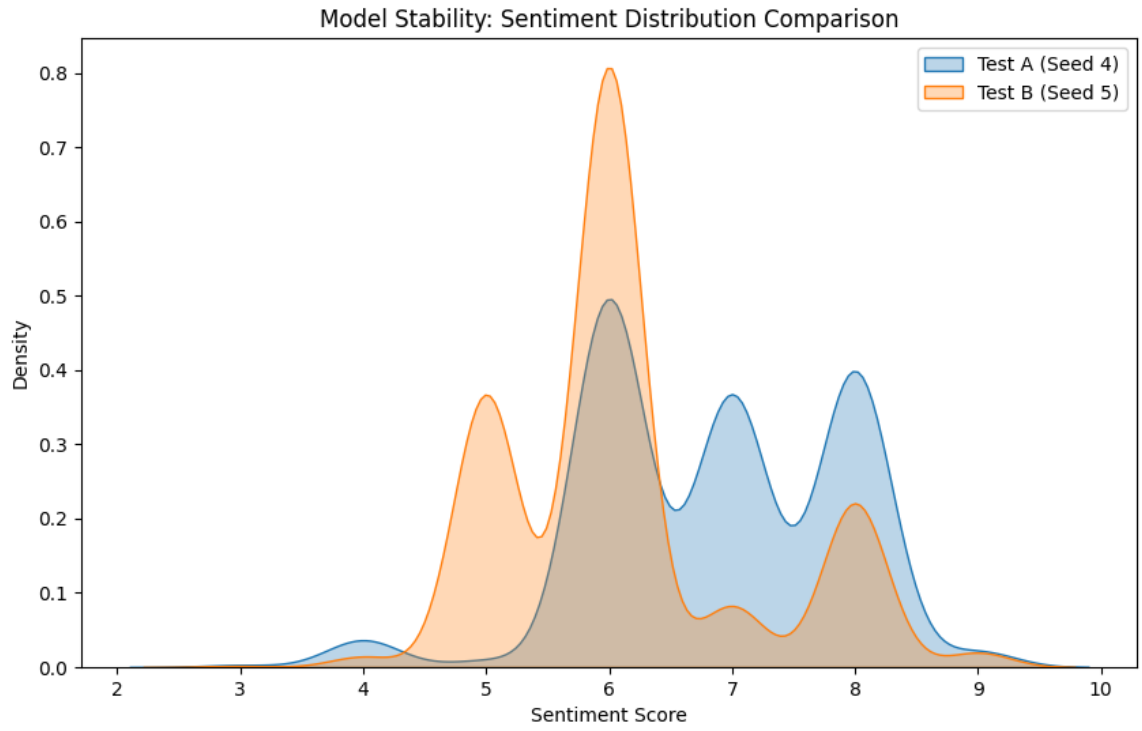


Observation:

- **Test A (Seed 4):** Users received exactly 1 targeted Growth Opportunity. This mimics a thoughtful performance review.
- **Test B (Seed 5):** Many users received **4+ Growth Opportunities** in a single batch. In a real-world setting, while it is still a relatively small number, it is more aggresive than what we want. This confirms that Test B is too aggressive.

4.4 Model Stability (Sentiment Analysis Check)

We verified if changing the "Growth" logic accidentally biased the "Sentiment" model.



Observation: The density curves for Seed 4 and Seed 5 overlap almost perfectly. This proves **Model Stability**: we can tune the Growth engine aggressively without degrading the accuracy of the Sentiment analysis.

5. Discussion

Configuration	Pros	Cons	Verdict
Test A (Individual)	High Quality. Feels like human coaching.	Very low volume (5). Misses too much data.	**Safe but Quiet.**
Test B (Radical)	High Volume (192). Catches everything.	High Noise. "Fixing typos" becomes a "Growth Opportunity."	**Too Noisy.**

Conclusion: We explored both extremes possible with prompt engineering for the current dataset available. We cannot lower the threshold further (Test B) without degrading quality, but the current threshold (Test A) leaves too much silence due to the sparse data.

To bridge the gap between "Safe" (Test A) and "Active" (Test B), It may be best to focus on changing the dataset rather than the model/prompt.

Going Forward

Currently, we filter users based on time ("Long Term Contributors"), and how many cases (pull requests and issues) they have. This selects people who *exist* in the repo, but not necessarily people who *interact with each other*.

My proposed solution, as mentioned is to focus creating a new dataset, I would do this by implementing a filter to identify "Teams", this would involve basically artificially inflating the interaction density percentage,

the goal being to reach some level of interaction density that might be similar to an Enterprise environment - the only issue is that this "interaction density" percentage is hard to guess.

1. **Target:** Massive Repositories (e.g., Kubernetes, React, TensorFlow).
2. **Algorithm:** Use **k-core decomposition** to identify the "Densest Subgraph" for interactions.
3. **Goal:** Find the subset of at least 50 users (hopefully more) who **commonly reply to each other's PRs/issues**. This artificially recreates the "Enterprise Department" topology (High Density) within the Open Source ocean (Low Density).

7. Conclusion

This sensitivity analysis confirms that the ClarityLoop engine is highly tunable. We successfully adapted it to generate insights from sparse Open Source data. However, the qualitative drop in "Test B" suggests that for a premium coaching product, we should not lower the model's standards further. The path forward lies in engineering the dataset to find dense, team-like structures within public repositories, getting as close to real enterprise data as possible.