Evaluating the Evaluators: Benchmarking Alignment Strategies for LLM-as-Judges

Shahul Es

Exploding Gradients
shahul@explodinggradients.com

August 18, 2025

Abstract

We study the effectiveness of large language models (LLMs) as automated evaluators, also known as LLM-as-judge, for real-world AI applications. Previous work often uses academic QA benchmarks that fail to capture the complexity of production settings. We create a synthetic, human-verified dataset containing domain-specific questions, expert grading notes, model responses, and expert verdicts.

Using this dataset, we benchmark leading proprietary models from OpenAI, Anthropic, and Google, and examine several in-context learning strategies including fixed few-shot, random few-shot, dynamic retrieval-based few-shot, and automatic prompt optimisation. Our results show that no single method consistently improves performance across all models, smaller distilled models respond unpredictably to optimisation, and larger reasoning-focused models benefit the most.

All code and data are available at https://github.com/explodinggradients/EvalsBench.

1 Introduction

Large language models are increasingly used as judges [1], serving as automated graders for AI-generated outputs in evaluation pipelines. This approach is now common in assessing Retrieval-Augmented Generation systems, agentic workflows, and other generative applications.

Typical evaluation dimensions include correctness, which checks whether the output contains the expected content; faithfulness, which measures whether the output is factually supported by provided context; and goal accuracy, which determines if the system achieved its intended outcome [2, 3, 4]. In most cases, evaluation involves a single model call that takes the system's output along with optional references such as ground truth answers or grading notes [5].

Although several studies and practical deployments show that LLMs can be effective judges, the current body of work has key limitations. Existing research heavily relies on public QA datasets such as TriviaQA [6], which contain short, single-word answers that do not represent the richness and variability of real-world responses. Many of these datasets are also likely to have been seen during model pretraining, creating potential contamination. Furthermore, there has been limited exploration of how prompt design and in-context learning strategies affect LLM judge performance across different models.

To address these shortcomings, this work makes the following contributions:

1. Creation of a synthetic but human-verified data set of 184 samples that better reflects realistic evaluation scenarios.
2. Benchmarking of LLM-as-judge performance across multiple proprietary models from OpenAI, Anthropic, and Google.
3. Systematic comparison of in-context learning approaches, including fixed examples, random selection, dynamic retrieval, and automatically optimised prompts.
4. Public release of all code and data to enable replication and further research.

2 Dataset

Most existing datasets for benchmarking LLM-as-judge do not reflect how these models are used in real-world evaluation settings. For example, TriviaQA, a widely used benchmark, contains many short, single-word answers such as:

> *question: Do You Know Where You're Going To? was the theme from which film?*
> *answer: mahogany*

Such formats are far removed from the longer, more nuanced responses seen in production AI systems, where outputs must address multiple requirements or reasoning steps.

To create a benchmark that better mirrors practical use cases, we designed and implemented a data generation and verification pipeline. For this study, the dataset contains 184 samples, each consisting of a question, expert-provided grading notes listing the expected points in an ideal answer, an LLM-generated response, and a verdict from a domain expert indicating whether the response meets the grading criteria.

The dataset was generated synthetically using the following multi-step pipeline:

1. Determine a topic and desired number of samples N to generate (e.g., "startup ecosystem" with $N = 200$).
2. Use an LLM to generate N subtopics for the given topic.
3. For each subtopic, create a question and grading notes outlining the required answer points.
4. Produce a high-quality response by providing the question and grading notes to an LLM.
5. Create a lower-quality response by removing one or more important points from the high-quality version.
6. Compile these into candidate samples.
7. Conduct manual verification by one or more human annotators to ensure that responses and verdicts are correct.

Below is an example sample from the dataset:

> *question:* What are the key methods for determining the pre-money valuation of a tech startup before a Series A investment round, and how do they differ?
>
> *grading notes:* DCF method: *future cash flows, requires projections; Comparable analysis: similar company multiples; VC method: revenue × multiple - post-$; Founder's share matters; strategic buyers pay more.
>
> *response:* Determining the pre-money valuation of a tech startup before a Series A investment round is a critical step, as it significantly influences the negotiation process and the ultimate percentage of ownership acquired by new and existing shareholders. Here's an overview of key valuation methods and their differences:
> 1. Discounted Cash Flow (DCF) Method: Forecast future cash flows, discount them to present value. Requires detailed projections and market justification.
> 2. Comparable Company Analysis: Compare with similar companies; use multiples like EV/Revenue.
> 3. Venture Capital Method: Estimate exit value, divide by expected ROI, work backwards.
> 4. Strategic Buyers' Perspective: Strategic acquirers may pay more due to synergies and market access.
>
> *verdict:* pass
> *reason:* response contains all points mentioned in the grading notes.

The dataset contains questions and responses related to the startup ecosystem, with each response averaging around 400 words, making it closer in length and complexity to real-world QA systems built using AI.

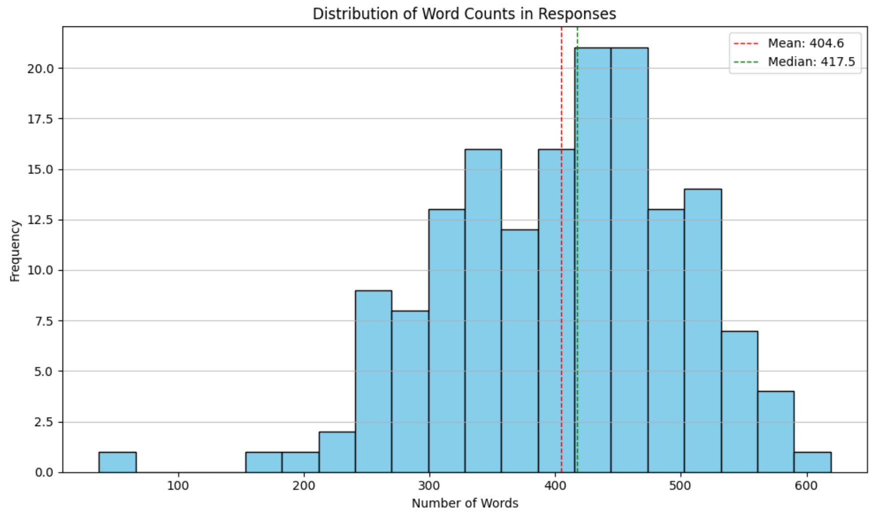
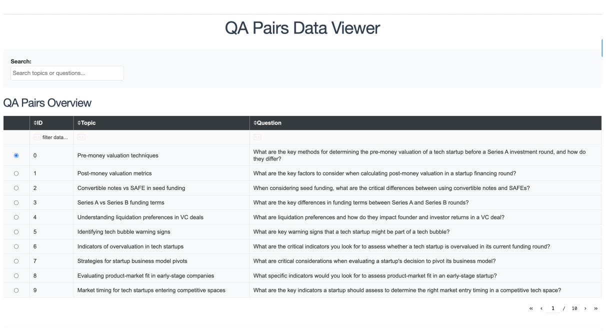
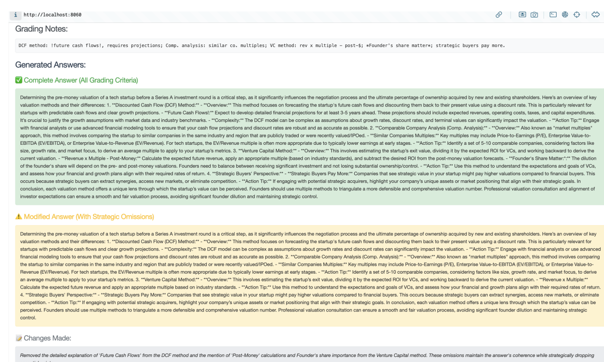


Figure 1: Distribution of number of word counts in responses of generated dataset.

We developed a custom web-based annotation interface to review and grade responses. Figure 2 shows two key views: (a) selecting and filtering QA pairs, and (b) reviewing generated answers alongside grading notes.



(a) QA Pairs Data Viewer interface.



(b) Generated answer review and annotation interface.

Figure 2: Annotation UI used for grading responses, showing question selection (left) and answer evaluation (right).

3 Methodology

Our goal is to measure and improve the alignment of large language models (LLMs) when used as automated evaluators, or "LLM-as-judges," against human-verified verdicts. We adopt a two-stage methodology: (1) establishing a vanilla evaluation baseline, and (2) applying in-context learning strategies to optimise judge performance.

Each evaluation sample consists of:
(i) a *question* from the startup ecosystem domain
(ii) expert-written *grading notes* listing key points expected in an ideal answer.
(iii) an *LLM-generated response*

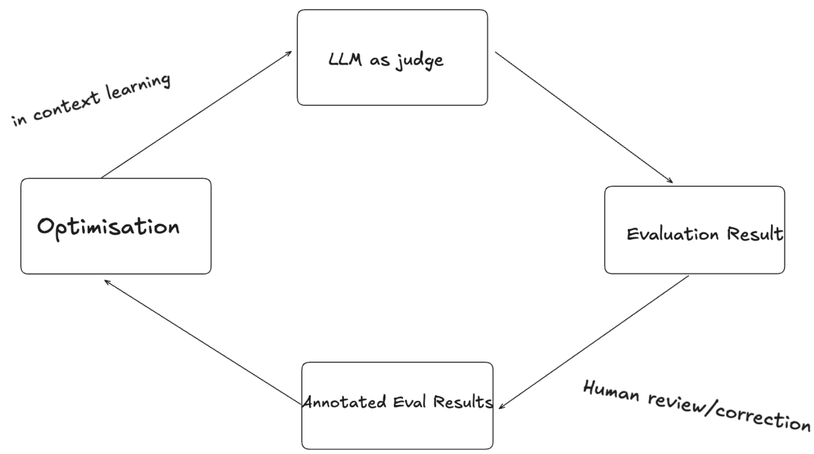


Figure 3: Evaluation and optimisation workflow for LLM-as-judges.

(iv) a binary *human verdict* (`pass` or `fail`) indicating whether the response meets the grading notes. The judge model receives the question, grading notes, and response, and outputs its own pass/fail verdict. Performance is measured using macro-averaged F1 score to equally weight both classes, avoiding bias from class imbalance.

We evaluate nine proprietary LLMs across three providers, accessed via their official APIs with temperature fixed at 0 for deterministic behaviour:

Provider	**Model**
Anthropic	claude-3.5-haiku
Anthropic	claude-4-opus
Anthropic	claude-4-sonnet
Google	gemini-2.5-flash
Google	gemini-2.5-flash-lite
Google	gemini-2.5-pro
OpenAI	gpt-4o
OpenAI	gpt-4o-mini
OpenAI	o3-mini

Table 1: Judge models evaluated. All runs used temperature = 0.

We first evaluate each judge model using a *vanilla prompt* containing the question, grading notes, and response, without any additional optimisation. This establishes the baseline F1 score for each model. To improve performance over this baseline, we created a small optimisation set of 34 samples from the dataset. A human evaluator reviewed the model's predictions on this subset, correcting any incorrect verdicts to produce a set of high-quality annotated examples. These serve as the basis for the in-context learning strategies described below.

Each optimisation strategy was applied to the remaining 150 samples from the dataset, with performance measured using the same macro-averaged F1 score as in the baseline evaluation.

Among the many in-context learning strategies proposed in prior work [7, 8], we focus on evaluating four commonly used optimisation approaches:

1. Fixed Few-Shot: A constant set of N annotated examples ($N = 2$) appended to every prompt, identical across all evaluations.
2. Random Few-Shot: For each evaluation, a random selection of N examples is drawn from the annotated pool and appended to the prompt.
3. Dynamic Few-Shot: The N most similar examples to the current evaluation sample are retrieved using cosine similarity over vectorised inputs (question and grading notes).
4. Automatic Prompt Optimisation: An LLM based coding agent (here Claude Code) analyses the annotated dataset and the original prompt, and proposes an optimised prompt aimed at maximising agreement with human judgments.

4 Experiments

We compare four in-context learning strategies for optimising LLM-as-judges against a vanilla baseline. All evaluations use the same dataset split described in Section 3, with macro-averaged F1 score as the performance metric. The nine models evaluated are listed in Table 1.

4.1 Baseline: Vanilla LLM-as-Judge

We first evaluate each model with a simple "vanilla" prompt, without any in-context examples or optimisation:

```
Check if the response contains points mentioned from
the grading notes and return 'pass' or 'fail'.
response: {response}
grading_notes: {grading_notes}
```

Results in 2 show that while larger models generally perform better than smaller ones, the pattern is not absolute. For example, Gemini 2.5 Flash-Lite slightly outperforms GPT-4o-mini despite having fewer parameters. None of the models achieve an F1 score above 90%, indicating substantial room for improvement.

4.2 Fixed Few-Shot

Two annotated examples from the human-reviewed set are appended to every evaluation prompt. These examples remain constant for all test samples, providing a consistent reference for pass/fail judgment.

As shown in 2, fixed few-shot yields modest improvements over the vanilla baseline for larger reasoning-focused models (e.g., Claude Opus, Gemini 2.5 Pro), but smaller or distilled models often degrade in performance. This suggests that static examples can be beneficial for high-capacity models but may constrain models with limited context-handling ability.

4.3 Random Few-Shot

Two annotated examples are drawn at random from the optimisation set for each evaluation. This introduces variety across prompts, with the aim of improving generalisation.

Results from 2 shows that random few-shot often outperforms fixed few-shot for adaptable models such as GPT-4o and Claude Opus, but results are inconsistent. Some models show negligible or negative changes, indicating that variability in context can sometimes reduce reliability.

4.4 Dynamic Few-Shot

Two most similar examples to the evaluation sample are retrieved from the optimisation set using cosine similarity over vectorised (question, grading notes) pairs. This ensures topical relevance between in-context examples and the evaluation task.

While this retrieval strategy is intuitively appealing, results from 2 shows that improvements are again concentrated among the largest models, with Claude Opus outperforming all others, followed by Gemini 2.5 Pro and o3-mini. Smaller models gain little or no benefit.

4.5 Automatic Prompt Optimisation

We employ an automated coding agent (Claude Code) to propose an improved evaluation prompt 6. The agent analyses the annotated dataset and the original vanilla prompt, then outputs an optimised version without adding examples. The task specification is shown below:

```
Act as prompt optimizer using human annotated data
data/annotation_df.csv. Optimise the instruction based
on this data (DO NOT ADD EXAMPLES) - src/prompt.txt.
Suggest a new prompt.
```

The new prompt 6 was then used to re-run evaluations across all models. As seen from 2, most models benefit substantially from the optimised prompt. However, smaller models such as GPT-4o-mini and Gemini 2.5 Flash-Lite degrade sharply, adopting a stricter judgment style that reduces alignment with human evaluators. This highlights that prompt designs optimised for high-capacity models may not directly transfer to smaller ones.

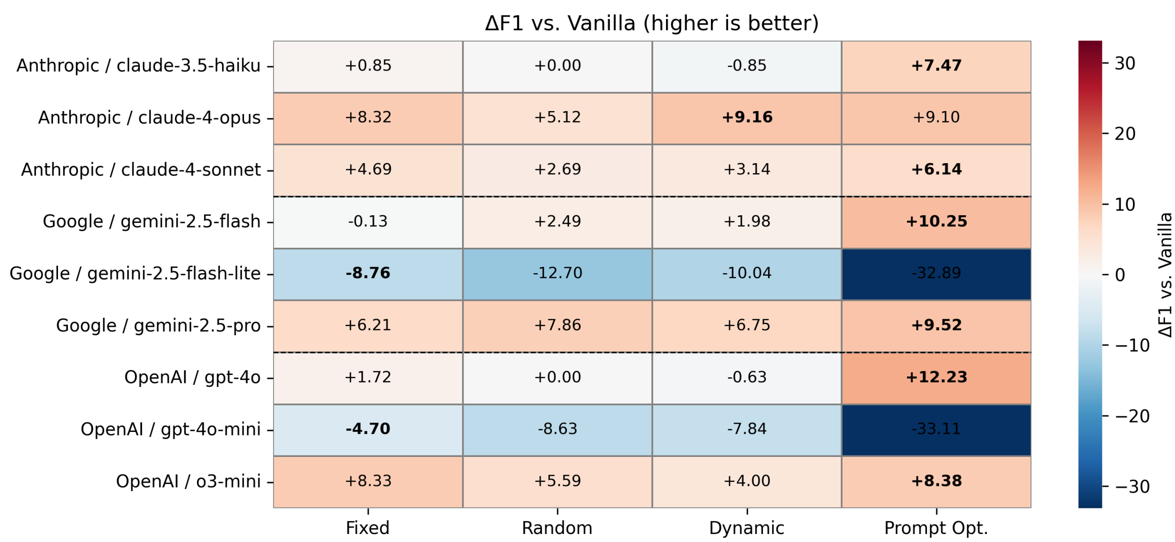


Figure 4: Change in macro-F1 relative to the vanilla baseline (Delta F1). Rows are models grouped by provider; columns are optimisation methods. Bold annotations mark the best method per model.

Table 2: Macro-averaged F1 scores for each model across the vanilla baseline and four optimisation strategies.

Provider	Model	Vanilla	Fixed	Random	Dynamic	Prompt Opt.
Anthropic	claude-3.5-haiku	66.95	67.80	66.95	66.10	74.42
Anthropic	claude-4-opus	86.02	94.34	91.14	95.18	95.12
Anthropic	claude-4-sonnet	87.43	92.12	90.12	90.57	93.57
Google	gemini-2.5-flash	87.91	87.78	90.40	89.89	98.16
Google	gemini-2.5-flash-lite	89.53	80.77	76.83	79.49	56.64
Google	gemini-2.5-pro	86.19	92.40	94.05	92.94	95.71
OpenAI	gpt-4o	80.63	82.35	80.63	80.00	92.86
OpenAI	gpt-4o-mini	84.49	79.79	75.86	76.65	51.38
OpenAI	o3-mini	87.43	95.76	93.02	91.43	95.81

5 Discussion

Our results reveal several key patterns about LLM-as-judge optimisation:

- **No universal best method:** No single optimisation method consistently outperforms others across all models. The impact of in-context learning appears closely tied to the underlying model capacity.
- **High model sensitivity:** LLM-as-judge performance is highly sensitive to the choice of base model. The same prompt and in-context strategy can produce very different alignments with human judgment.
- **Larger "thinking" models benefit most:** Models such as `o3-mini`, `Claude Opus`, and `Gemini 2.5 Pro` show an average performance increase of around 10% with the best-performing optimisation methods. This suggests that higher-capacity models are more steerable toward human expectations when given well-chosen examples or refined prompts.
- **Smaller distilled models behave unpredictably:** Models like `GPT-4o-mini` and `Gemini 2.5 Flash-Lite` often react counter-intuitively to common in-context learning approaches. This behaviour may stem from post-training or distillation processes applied to these models. 6
- **Anthropic models respond predictably to optimisation:** All Anthropic models tested improved in ways consistent with intuition. For example, consistently showing better performance than vanilla when using advanced optimization methods like dynamic few shot retrieval instead of fixed few shot.

6 Conclusion and Future Scope

LLM as judge metrics have emerged as an effective way to scale domain expertise when evaluating AI applications. In this study, we examined the effectiveness of one of the most common LLM as judge tasks, answer correctness, and evaluated how different in-context learning strategies can improve alignment with human evaluators. Our results show that optimisation strategies have highly model-dependent effects, with larger reasoning-focused models benefiting the most.

For future work, we plan to expand the scope of evaluation beyond correctness to include other commonly used metrics such as faithfulness. We also aim to increase both the size and diversity of the dataset to better reflect a wider range of real-world scenarios. In addition, we intend to explore advanced optimisation approaches proposed in recent literature like DSPy [9], PhaseEvo [10], etc to experiment if smaller models like gpt-4o-mini, etc can be tuned to behave like judges powered by larger model.

References

[1] Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*, 2024.

[2] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023.

[3] Sher Badshah and Hassan Sajjad. Reference-guided verdict: Llms-as-judges in automatic evaluation of free-form text. *arXiv preprint arXiv:2408.09235*, 2024.

[4] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.

[5] Yi Liu, Matei Zaharia, and Ritendra Datta. Enhancing llm-as-a-judge with grading notes. `https://www.databricks.com/blog/enhancing-llm-as-a-judge-with-grading-notes`, 2024. Databricks Blog.

[6] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[7] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[8] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2023.

[9] 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. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

[10] Wendi Cui, Jiaxin Zhang, Zhuohang Li, Hao Sun, Damien Lopez, Kamalika Das, Bradley Malin, and Sricharan Kumar. Phaseevo: Towards unified in-context prompt optimization for large language models. 2024.

Appendix

Section A: Prompt proposed by claude code

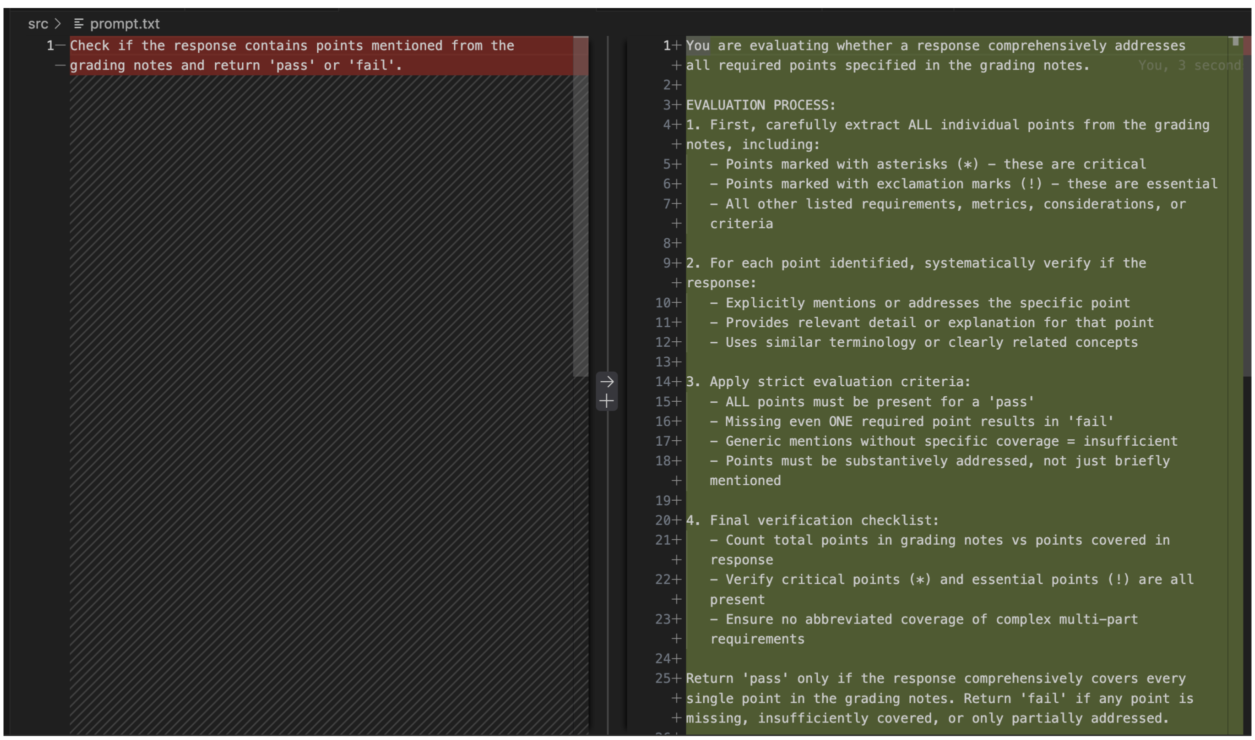


Figure 5: Git diff view between vanilla prompt and Claude Code–generated optimised evaluation prompt

Section B: Claude code's logs while optimising prompt

```
> Act as prompt optimizer using human annotated data data/annotation_df.csv.
  Optimise the instruction based on this data (DO NOT ADD EXAMPLES) -
  src/prompt.txt. Suggest a new prompt

[INFO] I'll help you optimize the prompt using the human annotated data. Let me
       first examine the current prompt and the annotation data to understand what
```

needs to be improved.

[INFO] Read(src/prompt.txt)
-> Read 3 lines (ctrl+r to expand)

[INFO] Read(data/annotation_df.csv)
-> Read 591 lines (ctrl+r to expand)

[INFO] Now I'll analyze the current prompt and annotation data to understand the patterns and create an optimized version.

Based on my analysis of the current prompt and the human annotated data, I can see clear patterns in what makes responses pass or fail. The current prompt is very basic and doesn't provide enough guidance for generating comprehensive responses that cover all required points.

[INFO] User approved Claude's plan:
-> Prompt Optimization Plan

Current Issues Identified:
- Current prompt is too generic and lacks specificity
- No guidance on comprehensive coverage of grading criteria
- Missing instructions on thoroughness and point verification
- No structure for systematic response generation

Key Patterns from Failed Responses:
- Missing specific required points (e.g., diversity stats, compliance costs, grants, IP ownership)
- Incomplete coverage of grading notes criteria
- Lack of systematic verification against all marked points

New Optimized Prompt Structure:
1. Clear Task Definition: Explicitly state the need to check ALL points in grading notes
2. Systematic Verification Process: Add step-by-step verification instructions
3. Emphasis on Completeness: Require comprehensive coverage of every marked point
4. Quality Assurance: Include self-checking mechanism before final response
5. Specific Formatting: Ensure response addresses each grading note systematically

Expected Improvements:
- Higher pass rate by ensuring all grading criteria are covered
- More systematic and thorough responses
- Better alignment between responses and evaluation criteria
- Reduced false negatives due to missing key points

The new prompt will transform the current basic instruction into a comprehensive evaluation framework that guides the model to systematically verify all required points before generating the final response.

[INFO] Update Todos

```
      -> [x] Analyze patterns from annotation data to understand pass/fail criteria
      -> [x] Identify key issues with current prompt based on failed responses
      -> [ ] Design new optimized prompt structure with systematic verification

[INFO] Update(src/prompt.txt)
      -> Updated src/prompt.txt with 26 additions and 1 removal
         1 -  Check if the response contains points mentioned from the grading notes and return 'pass'
         1 +  You are evaluating whether a response comprehensively addresses all required points spec
         2 +
         3 +  EVALUATION PROCESS:
         4 +  1. First, carefully extract ALL individual points from the grading notes, including:
         5 +     - Points marked with asterisks (*) - these are critical
         6 +     - Points marked with exclamation marks (!) - these are essential
         7 +     - All other listed requirements, metrics, considerations, or criteria
         8 +
         9 +  2. For each point identified, systematically verify if the response:
        10 +     - Explicitly mentions or addresses the specific point
        11 +     - Provides relevant detail or explanation for that point
        12 +     - Uses similar terminology or clearly related concepts
        13 +
        14 +  3. Apply strict evaluation criteria:
        15 +     - ALL points must be present for a 'pass'
        16 +     - Missing even ONE required point results in 'fail'
        17 +     - Generic mentions without specific coverage = insufficient
        18 +     - Points must be substantively addressed, not just briefly mentioned
        19 +
        20 +  4. Final verification checklist:
        21 +     - Count total points in grading notes vs points covered in response
        22 +     - Verify critical points (*) and essential points (!) are all present
        23 +     - Ensure no abbreviated coverage of complex multi-part requirements
        24 +
        25 +  Return 'pass' only if the response comprehensively covers every single point in the grad
        26 +  Return 'fail' if any point is missing, insufficiently covered, or only partially address
        27    response: {response}
        28    grading_notes: {grading_notes}

[INFO] Update Todos
      -> [x] Analyze patterns from annotation data to understand pass/fail criteria
      -> [x] Identify key issues with current prompt based on failed responses
      -> [x] Design new optimized prompt structure with systematic verification

[INFO] The optimized prompt transforms the basic instruction into a comprehensive
      evaluation framework with:

  1. Systematic Point Extraction: Explicitly instructs to identify ALL points
     including critical (*) and essential (!) markers
  2. Thorough Verification Process: Step-by-step checking of each point's
     coverage
  3. Strict Evaluation Criteria: Clear pass/fail thresholds requiring complete
     coverage
  4. Quality Assurance Checklist: Final verification steps to prevent missing
     points

  This addresses the main issues from failed responses: incomplete coverage,
```

```
missing critical points, and lack of systematic verification against grading
criteria.
```

Section C: Analysis of reasoning change with optimised prompt

Below examples shows how both models (gemini-2.5-flash-lite & gpt-4o-mini) changed its reasoning and verdict from "pass" to "fail" based on the new proposed prompt by claude agent. It seems to have overfitted the instruction and reasons very strictly about trivial aspects like "massive aspects" vs "excessive aspects" which leads to poor alignment.

Example 1

```
### Judge's reason with vanilla prompt

The response mentions 'Crazy Valuations and Revenue Multiples' with a 10x threshold, 'Unsustainable Bur
These points directly correlate with the grading notes.

### Judge's reasoning with optimised prompt

The response did not meet all the requirements.
Specifically, the grading notes mentioned 'customer acquisition massive spend' while the response only 
Additionally, the response did not explicitly state that the burn rate was 'unsustainable' but rather d
```

Example 2

```
### Judge's reason with vanilla prompt

The response mentions Series A funding details such as investment size ($2 million to $15 million), pos
It also covers Series B funding details including investment size ($7 million to over $30 million), gro

### Judge's reasoning with optimised prompt

The response fails to meet the requirements.
Specifically, for Series A, the 'post-money valuation' is mentioned but not elaborated upon with the ex
Additionally, the grading notes imply a need to discuss 'more dilution' for Series B, which is only gen
```

Example 3

```
### Judge's reason with vanilla prompt

The response successfully captures key points from the grading notes,
including the need for future cash flow projections in the DCF method, mention of similar company multi

### Judge's reasoning with optimised prompt

The response fails to comprehensively cover all required points from the grading notes.
Specifically, while it does address some aspects of each method, it does not sufficiently clarify or em
Key points like the detailed requirements for 'future cash flows' projections are outlined in general b
These omissions lead to a conclusion of a fail.
```