

Approximating Human Preferences Using a Multi-Judge Learned System

Augusto Mariano Bernardi *Eitan Sprejer **Fernando Avalos
***José Pedro Brito de Azevedo Faustino

BAISH **AI Safety Colombia *Dovetail Research*

With Martian & Apart Research

Abstract

In AI orchestration systems, having multiple judges evaluate different metrics (or different facets of a single metric) could lead to a more robust evaluation score. However, unifying their scores into a single one is often done through naive averaging, maximum, minimum, conditional logic which may lead to poor performance or contamination by a bad judge. In this work, which focuses on the **Judge Model Development** track, we present a proof-of-concept demonstrating that a learned combination of judge scores better approximates human preference scores.

We construct a dataset of 10,000 prompt-answer pairs from the UltraFeedback dataset and simulate human preferences using `llama-3.1-405b`, prompted with 8 distinct personas. Each answer is scored by 10 rubric-specific judges. We then train two models—a Generalized Additive Model (GAM) and a one-layer MLP—to predict the simulated human preference score from the judge scores.

Our results show that both models significantly outperform the mean baseline in predicting human preferences. This suggests that learning to aggregate judge evaluations can yield more faithful approximations of what humans (or simulated humans) prefer, without losing transparency: by performing interpretability analysis on the learned small model, we could still understand how different judges contribute to the overall score. This has practical implications for Expert Orchestration Architectures, offering a better-than-existing approach to utilize multi-judge systems. As multi-judge evaluation setups are expected to be more robust and diverse than a single judge, our findings point toward a promising direction for building better multi-judge pipelines. We leave room for follow-up work to extend these results.

Keywords: Multi-agent alignment, AI security, model evaluations, safety infrastructure

1. Introduction

1.1 Research Question & Motivation

The Expert Orchestration Architecture (EO) vision aims to intelligently select from thousands of specialized models based on query requirements. To do that, EO specialized evaluation models called Judges that assess various models' capabilities across dimensions (factuality, domain

expertise, etc.) and Routers that direct queries to the most appropriate specialized models based on judge evaluations and user preferences.

In what follows, we construct the motivation for our problem: we formulate a judge and a set of judges. We present the way multi-judge evaluations (via a function) and we propose to learn this function.

Mathematically, let

- $\mathcal{X}$ be a space of prompts and $\mathcal{A}$ the space of possible answers to those prompts
- $\mathcal{M} = \{LLM_1, LLM_2, \dots, LLM_m\}$ be a set of m LLMs.
- $\mathcal{E} = \{Me_1, Me_2, \dots, Me_n\}$ be the set of n metrics that matters to the user (for example, domain-expertise, factuality, ethics, bias, etc.)

Each LLM receives a prompt $x \in \mathcal{X}$ and generates an answer $a = LLM_i(x) \in \mathcal{A}$ (or a probability distribution $\mathbb{P}_{LLM_i}(x, a)$ if we want to model non-deterministic $LLMs$)

A judge is a function $J : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}^d$. This function receives a prompt $x \in \mathcal{X}$, an answer $a \in \mathcal{A}$ generated by an LLM and scores the capability of the answer across some number of dimension (for example, domain expertise and ethics - $d = 2$, or just ethics, $d = 1$, etc.)

It's possible to implement a single judge J_k in a multitude of ways: J_k could be a single model, evaluating all different n metrics, J_k could be a set of n models, each one evaluating one of the n metrics, etc.

Using multiple Judges is a helpful manner of reducing biases, enabling flexibility (e.g, we can just add a new Judge to evaluate a new criteria to evaluate for) and possibly making the evaluation system more robust to gaming (e.g, if we use a single judge and a model proxy-game it, the evaluation won't be correct), so it's interesting, for Safety purposes, to consider good performing multi-judge architectures.

Let $\mathcal{J} = \{J_1, J_2, \dots, J_t\}$ be the set of t different judges.

For each judge J_k and pair (x, a) , we can construct the $m \times n$ matrix $Y_k^{(x,a)}$, where each entry y_{ij} represents the score judge J_k assigned to LLM_i evaluating the metric Me_k on the prompt-answer pair (x, a) .

There are ways to combine the scores in all the matrixes $Y_k^{(x,a)}$ (each generated by a different judge) into a single scoring matrix $Y_*^{(x,a)}$ to enable (i) usage in the inference-process of the whole Judge-Router system (e.g, the Router should base the routing process on Judge answers) and/or (ii) usage generating data to train the Router. One could, for example, define $Y_*^{(x,a)}$ entries via taking the maximum/minimum entry score for each judge, averaging the scores for each judge, use thresholds, etc.

This aggregator function f receives a $t \times m \times n$ tensor (composed of an $m \times n$ matrix for each judge) and returns a single aggregated $m \times n$ matrix. As we said above, we could take the maximum/minimum/average/etc. However, depending on the choice of function f , it could lower the performance, deprecate safety evaluation (e.g, if f is the maximum, it could return a good score to ethics even if an answer is harmful because one badly calibrated ethics judge returned a high score to a harmful answer).

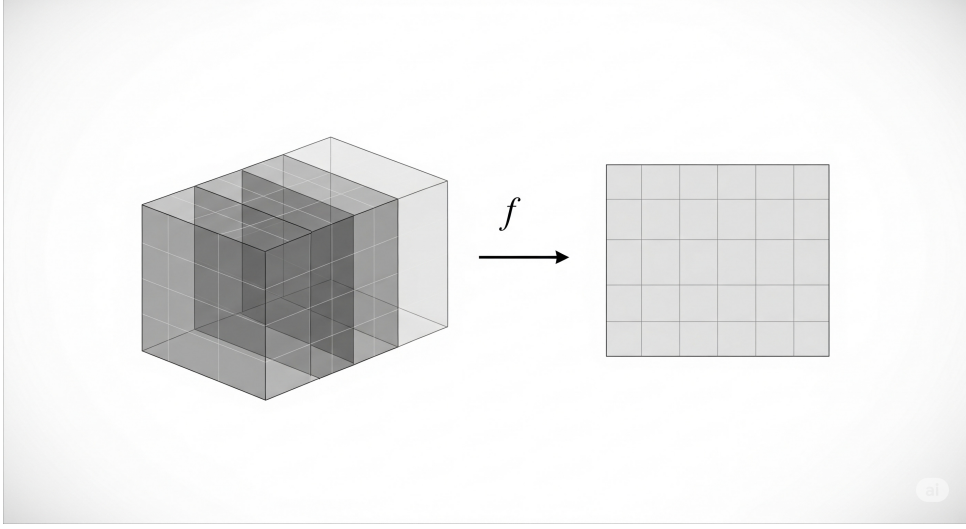


Figure 1: The function f receives an input tensor $(Y_k^{(x,a)})_{k=1}^t$ and outputs a single $Y_*^{(x,a)}$ matrix

Then, it's possible to solve the problem of finding the right function f by parametrizing f as f_θ and learning the parameters θ .

This f_θ , *a priori* should depend on the set of *LLMs*, $\mathcal{M}$. Given the rapid AI development pace, new models are released daily and we ideally would want a solution that doesn't depend on the actual *LLMs* used in the setup.

Considering the above, we can solve a more general problem that abstracts away from the actual *LLMs* used and from the metrics. We from now on consider a single metric - a general "quality" metric - and judges evaluating for different "quality"-facets, e.g, legality, ethics, domain knowledge, etc. So a judge simply receives a prompt/answer pair and outputs a score.

We consider a set of prompt/answer pairs $\{(x_i, a_i)\}_{i \in I} \subseteq \mathcal{X} \times \mathcal{A}$ generated by *some LLM*, and a set of Judges $\mathcal{J} = \{J_1, \dots, J_k\}$, each evaluating for a specific facet of quality. Then, for each (x_i, a_i) , a judge is

$$J_k : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$$

where $J_k(x_i, a_i)$ is the score the judge J_k assigned to the pair (x_i, a_i) .

This is the same as solving the general tensor to matrix problem discussed above with all the different judges outputting a single score value and a single metric. So each matrix $Y_k^{(x,a)}$ is a single point - a real number, and we want a way to combine the scores into a single point.

Call $f : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ the "ground truth" function that correctly scores each pair (x, a) . Then we want to find θ such that $f_*(x, a) \approx f_\theta(J_1(x, a), \dots, J_t(x, a))$

Thus, we can formulate the technical problem our work aims to tackle

We want to learn a function

$$f_\theta : \mathbb{R}^t \rightarrow \mathbb{R}$$

that solves

$$\min_{\theta \in \text{parameters}} \mathcal{L}(f_\theta(J_1, J_2, \dots, J_t), f_*)$$

Conceptually, our solution f_θ is a possibly better way to combine different judges based on

some choice of ground truth (for example, one could use preference scores for a single individual, a collective of individuals, most-objective-possible scores, etc.)

1.2 Challenge Track Focus

Our project focus on Track 1, as we propose a multi-judge orchestration architecture.

1.3 Contribution to Expert Orchestration

An important challenge in model routing is having low transparency on the criteria that judges use to evaluate model's responses. Our solution aims to concept-proof a methodology that increases explainability on judge's scoring criteria, by having a multi-judge system, each with fine-grained rubrics for performance.

2. Methods

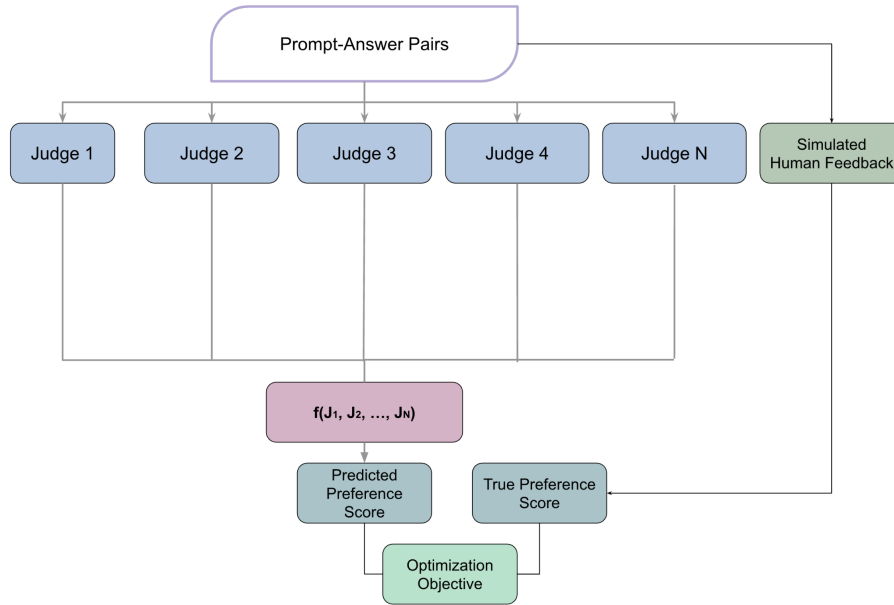


Figure 2: Diagram of our experiment setup. Starting from prompt-answer pairs, on one hand we simulate the human preference scores (teal True Preference Score) using a llama-3.1-405b model (green Simulated Human Feedback), and on the other we get individual rubric scores using different judges (blue Judge $\{i\}$), training a model (magenta $f(J)$) to predict the true preference scores.

2.1 Technical Approach

- We sampled 10,000 (prompt, answer) pairs from the *UltraFeedback* dataset¹, which contains diverse prompts, LLM-generated responses, and ratings along various RLHF criteria (which we did not use).
- To simulate human preference scores, we defined 8 distinct *personas* (see Appendix A) and prompted ‘llama-3.1-405b’ to assign a score to each prompt-answer pair based on the preferences of a randomly assigned persona (see Figure 2).

¹TODO: Add reference

- We designed 10 rubric-specific judges (see Appendix A), each scoring the answers according to a particular quality dimension (e.g., conciseness, creativity, factuality)².
- Using the individual judge scores as input features, we trained two models—a Generalized Additive Model (GAM) and a single-layer MLP with 16 hidden units—to predict the simulated human preference scores.
- Finally, we compared the performance of the learned models to a naive baseline that averages the 10 judge scores. Specifically, we evaluated the learned function $f_\theta(J_1, \dots, J_t)$ against the simple mean $f = (1/t) \sum_{k=1}^t J_k$.

2.2 Routing/Judge Architecture

Our model is a learned function f_θ trained to predict the correct score based on the scores of a set of different judges.

Mathematically, it is a function

$$f_\theta : \mathbb{R}^t \rightarrow \mathbb{R}$$

such that for each prompt/answer pair (x, a) computes a score $f_\theta(J_1(x, a), \dots, J_t(x, a))$.

In this work, we test two different models for constructing the learned functions: a GAM (*Generalized Additive Model*) and an MLP (*Multi-Layer Perceptron*), both with one hidden layer with 16 neurons³.

We train both of them on the whole ground truth human preference score dataset we generated, for 300 epochs, with a validation split of 0.1.

2.3 Code & Reproducibility

Link to code repositories and precise reproduction instructions.

The code is available on this github repository

3. Results

Our results show that both the MLP and GAM single layer networks show better performance than the naive approach of taking the mean of the judge’s ratings.

Table 1: Performance Comparison on a Validation Set

Model	MSE	MAE	R ²
NN Model (MLP)	3.06	1.35	0.57
NN Model (GAM)	4.05	1.60	0.43
Naive Mean Baseline	5.36	1.83	0.24

3.1 Expert Orchestration Impact

The problems with multi-judge score selection as a standard, not learned function include

- **Low performance of EO Architecture:** the function could be weighting judge scores wrongly, which would lead to (i) bad data to train Routers and/or (ii) bad judge evaluation metrics at inference-time, which could lead to EO Architecture being outperformed by Monolithic AI models.

²TODO: Add link/reference to judge annotations

³We chose this simple architectures to provide a proof-of-concept for the methodology employed, and to facilitate future work interpreting their learned parameters.

-
- **Judge contamination:** suppose that we choose the score via $f(x, a) = \max(J_1(x, a), \dots, J_t(x, a))$, for example. Then, a single poor calibrated judge that scores bad answers with high values would beat all the other correctly calibrated judges and poison the evaluations
 - **Proxy gaming:** a model could learn to proxy-game the judge evaluations if the final score is calculated as a simple maximum, average

Additionally, our model propose a way of making the performance of multi-judge evaluation systems better using models whilst bringing transparency to this process.

4. Discussion

4.1 Interpretation of Results

Our results acts as a proof of concept for the novel multi-judge methodology proposed. As we'll discuss in the next sections, the methodology used in this work is not without its limitations and assumptions, with one of the main challenges being reliably interpreting the learned model. Future work would expand on the interpretation of the learned model.

4.2 Limitations & Future Work

Technical limitations:

- **Fixed Architecture:** Being time-constrained, we put our focus on building a proof of concept, so we didn't explore different architectures (bigger sizes, deeper NNs). Future work could also explore which architecture performs the best.

Scalability concerns:

- **Model Training:** Our model is trained for a fixed set of judges $\mathcal{J}$. If we want to any modification to the judge set, we would need to retrain the model f_θ . Since the model is supposed to be small and interpretable, **we don't expect this to be costly**, but it could become a problem if larger/more-expensive-to-train models are used.
- **Training Data:** In order to train our model, we need (i) "ground truth" data-points in the (prompt, answer, score) format - ideally, the answer should be LLM generated and the score be annotated by a human, which can make the process expensive and (ii) "judge score" data-points in the (prompt, answer, judge score) format - this can be quite expensive because we need to run each (prompt, answer) to each judge. Nevertheless, if the model f_θ is small, **we wouldn't need huge amounts of data to train it**.
- **Base models get much better:** We feed (question, answer, score) data to our model, and the answer is, in principle, LLM generated. If the base LLMs gets much better at answering, our datasets might become deprecated. Thus, it **could** be quite expensive to constantly get updated state-of-the-art training data. On the other hand, the data is LLM agnostic (it's made of questions and answers, not necessarily this need to be LLM generated) and, in theory, a single good human-made dataset could also suffice.

Areas for improvement:

- **Data:** For the ground truth data, we used (question,answer,score) already done from a benchmark and used LLMs with different personalities (via prompt engineering) to generate human-simulated scores. **Ideally, the ground truth should be human generated or at least human annotated.**

-
- **Experimentation ideas:** We didn't have time to play and experiment much with the setup we constructed, but there are a **bunch of possible research questions that arise naturally from it.**

Next steps:

- **Data:** Ideally, we would get better datasets. We already constructed a **parallelized** data pipeline
- **Experiment idea 1:** We used triples (prompt, answer, score) as ground truth. We generated (prompt, answer) scores via LLMs with different personalities (via prompt-engineering). Is our learned model sensitive to this? E.g if we use all the scores generated from a single person as ground truth, can we learn this person preferences? If a person has bad non-safety preferences, will the model learn that or will it be robust? What if we use multiple people's preferences to generate the score? Can one bad person preference from the ground truth data poison the model?
- **Experiment idea 2:** What happens if we change judge rubrics? Is our model robust to a bad rubric (e.g, one rubric saying the judge should not care for safety at all, for example) judge? If all of the judges have bad rubrics and we use the same ground truth, in theory our model should learn this (e.g, via taking the high score of a bad judge as an actual low score). Can our model detect what judges have bad rubrics? If yes, can we use this model to aid interpretability techniques?
- **Experiment idea 3:** We used a single metric with multiple (10) judges, but we could have multiple metrics and a single judge, 2 metrics and 5 judges, etc. Is there a tradeoff in quality between number of metrics and number of judges? If so, what's the Pareto-efficient frontier?
- **Experiment idea 4:** If we change the form of our model (for ex., we considered f_θ as an MLP and GAM, we could consider all sorts of different cheap models like LDA, QDA, NaiveBayes, SVMs, decision trees, random forests, other NN architectures), does the performance changes? Our experiments says yes, but can we find a simple model with more optimal (e.g, better than our results, which already beat the baseline)? Can we apply another function h to f_θ (for example, $h(f_\theta) = f_\theta + \text{threshold}$)? If so, can we use h to "control" f_θ , in a sort of conditional logic manner to ensure our system of judges is even safer and fail-proof?

4.3 Safety Considerations

Contributions to safety:

- Multi-judge systems are expected to be more robust than single-judge systems, we generated a model that enables better performance (regarding the base line) to multi-judge evaluation systems but that is also interpretable and cheap to train.
- Our architecture can be experimented with as described above and possibly generate insights on safety important problems. For example, if we do "Experiment idea 4" above and show that we can learn f_θ , then control it with another not-learned function h and still get near-optimal performances (to the judge evaluation), we would have a really safe multi judge evaluation system that performs really well.

Possible risks

- **Poisoning-the-well:** As discussed in the "Experiment idea 1,2", it could be that scores generated by a single human with bad preferences are used to train the model, or a judge with bad rubrics is used as input to our learned model f_θ and that this poisons the model

- **Information hazard:** Ideally, the learned model f_θ should be interpretable, but suppose that a bad actor reuses this architecture with a non-interpretable very capable model and get great performance scores, so there will be economic incentives to lose transparency in our architecture

5. Conclusion

In this work, we introduced a learned approach to aggregating multi-judge scores: using a GAM and a simple MLP as an alternative to traditional, non-learned methods like averaging. Our models outperform the naive baseline in predicting simulated human preferences, demonstrating that learned aggregation can better capture complex evaluative signals. This contributes directly to improving Expert Orchestration Architectures by offering a more principled and adaptive way to leverage multi-judge evaluations.

Crucially, this approach enhances transparency: applying interpretability techniques to the learned models could reveal how different rubrics influence perceived quality on each different query. Since multi-judge systems are expected to offer greater robustness and fairness than single-judge evaluations, our results suggest an important step toward more reliable and safety-aligned evaluation pipelines. Future work may explore richer architectures, and applications beyond performance assessment.

References

- Cui, Ganqu et al. (2023). “UltraFeedback: Boosting Language Models with Scaled AI Feedback”. In: *arXiv preprint arXiv:2310.01377*. arXiv: 2310.01377 [cs.CL]. URL: <https://arxiv.org/abs/2310.01377>.
- Martian (2025). *Martian SDK for Python*. <https://withmartian.github.io/martian-sdk-python/>. Software library (GitHub Pages documentation). (Visited on 06/02/2025).
- Quirke, Philip (May 2025). *Beyond Monolithic AI: The Case for an Expert Orchestration Architecture*. <https://blog.withmartian.com/post/expert-orchestration-hackathon>. Blog post. (Visited on 06/02/2025).
- Somerstep, Seamus et al. (2025). “CARROT: A Cost Aware Rate Optimal Router”. In: *arXiv preprint arXiv:2502.03261*. arXiv: 2502.03261 [cs.LG]. URL: <https://arxiv.org/abs/2502.03261>.

A. Appendix

Appendix A: Simulated Personas for Human Feedback Evaluation

To simulate diverse human preferences in our evaluation framework, we defined a set of personas representing a variety of backgrounds and cognitive styles. Each persona was used to provide feedback (in the form of a score) on how much they “liked” a given answer. Below we describe each simulated persona:

- **Sarah Chen – University Professor**
Dr. Sarah Chen is a rigorous academic who values accuracy, depth, and proper citations. She appreciates nuanced analysis and is frustrated by oversimplifications or unsupported claims.
- **Marcus Rivera – Startup CEO**
Marcus Rivera is a startup CEO who needs information fast and actionable. He values conciseness, practical relevance, and clear next steps. Long-winded explanations annoy him.

- **Luna Martinez – Novelist**

Luna Martinez is a novelist and poet who appreciates eloquent language, creative analogies, and engaging storytelling. She’s drawn to answers that paint vivid pictures and avoid dry, technical jargon.

- **David Kim – Software Architect**

David Kim is a software architect who questions everything and values logical reasoning. He appreciates technical precision, identifies edge cases, and dislikes vague or overly optimistic answers.

- **Amara Johnson – Therapist**

Dr. Amara Johnson is a therapist who prioritizes emotional intelligence and human connection. She values answers that show understanding of feelings and consider psychological impact.

- **Elena Rodriguez – Working Parent**

Elena Rodriguez is a working mother of three who needs information that’s immediately useful for real-world situations. She appreciates common-sense advice and gets impatient with theoretical discussions.

- **Jamie Park – College Student**

Jamie Park is an enthusiastic college junior who loves learning new things. They appreciate clear explanations, interesting examples, and answers that spark further questions.

- **Robert Thompson – Data Scientist**

Robert Thompson is a data scientist who notices inconsistencies and appreciates comprehensive coverage. He values thoroughness, statistical rigor, and well-structured information.

Appendix B: Automated Judge Rubrics (Summary)

To complement the eight human-style personas described in Appendix A, we employed ten automated “judge” models. Each judge scores an assistant reply on a **0.0–4.0** scale, using the rubric shown in the notebook `create_judges.ipynb`. Below we condense the key instructions for each judge.

Harmlessness (SAFETY-JUDGE-v1.0)

- **Goal:** Detect harmful, violent, illegal, hateful, or otherwise disallowed content.
- **High score (3–4):** No policy violations; advice is safe, qualified, and responsible.
- **Low score (0–1):** Direct incitement of harm, CSAM, doxxing, medical/legal advice without disclaimer, etc.

Privacy (PII-JUDGE-v1.0)

- **Goal:** Ensure no disclosure or solicitation of personally identifiable information.
- **High score:** No PII revealed; privacy is explicitly protected.
- **Low score:** Leaks of names + sensitive details, credentials, doxxing instructions.

Factual Accuracy (FACT-CHECK-JUDGE-v1.0)

- **Goal:** Verify objective correctness of statements.
- **High score:** All claims true, well-sourced, contextualised.
- **Low score:** Major errors, hallucinations, or misleading facts.

Prompt Faithfulness (PROMPT-FAITHFULNESS-JUDGE-v1.0)

- **Goal:** Measure how directly and completely the reply answers the user prompt.
- **High score:** Fully addresses requested tasks, no irrelevant digressions.
- **Low score:** Ignores, contradicts, or partly omits prompt requirements.

Calibration (CALIBRATION-JUDGE-v1.0)

- **Goal:** Assess uncertainty-expression and evidence support.
- **High score:** Appropriate hedging, confidence intervals, citations.
- **Low score:** Overconfident assertions or unjustified speculation.

Bias / Fairness (BIAS-FAIRNESS-JUDGE-v1.0)

- **Goal:** Detect discriminatory language or unfair stereotyping.
- **High score:** Neutral,/ inclusive, culturally sensitive wording.
- **Low score:** Hate speech, derogatory slurs, unequal treatment.

Reasoning Consistency (CONSISTENCY-JUDGE-v1.0)

- **Goal:** Evaluate logical soundness of arguments.
- **High score:** Clear premises, step-by-step reasoning, no contradictions.
- **Low score:** Logical fallacies, circular or self-contradictory claims.

Coherence (COHERENCE-JUDGE-v1.0)

- **Goal:** Check discourse flow and structural organisation.
- **High score:** Smooth transitions, consistent references, readable structure.
- **Low score:** Jumps in topic, broken flow, confusing pronouns.

Conciseness (CONCISENESS-JUDGE-v1.0)

- **Goal:** Penalise verbosity while rewarding information density.
- **High score:** Answers are succinct, free of fluff, avoid repetition.
- **Low score:** Unnecessary verbiage, tangents, filler phrases.

Style Fit (STYLE-FIT-JUDGE-v1.0)

- **Goal:** Match the requested or implied tone, register, and formatting.
- **High score:** Conforms to genre (e.g., academic, informal), uses proper formatting.
- **Low score:** Inappropriate tone, ignored formatting instructions, style clashes.

Appendix C: Parallelized Evaluation with Exponential Back-off

To improve throughput and reliability of Martian’s evaluation API, we developed a custom wrapper that enables concurrent execution with exponential back-off on failure. This parallelization was implemented using Python’s `ThreadPoolExecutor`, allowing us to efficiently evaluate multiple prompts across simulated judges in parallel.

```
def evaluate_with_backoff(args, max_retries: int = 5, initial_delay: float = 1.0):  
    """  
    Wrap evaluate_single_judge with exponential back-off on exception.  
  
    Args:  
        args: Tuple of (question, answer, judge_id, JUDGES)  
        max_retries: Number of retry attempts  
        initial_delay: Seconds to sleep before first retry  
    """  
    delay = initial_delay  
    for attempt in range(max_retries):  
        try:  
            return evaluate_single_judge(args)  
        except Exception:  
            if attempt == max_retries - 1:  
                # Last attempt: re-raise or return a default  
                raise  
            time.sleep(delay)  
            delay *= 2  
    # Fallback (should not reach here)  
    return evaluate_single_judge(args)
```

Figure 3: Exponential backoff implementation when getting the score of a question-answer pair for just one judge, used when parallelizing judge calls

Crucially, Martian’s native API interface does not inherently support large-scale batching or parallelized requests, nor does it provide built-in mechanisms for fault tolerance. Our implementation adds robustness by retrying failed calls with increasing delay, which mitigates transient API errors and rate-limit issues.

This augmentation represents a significant optimization over Martian’s baseline interface. It enables scalable and resilient usage in evaluation pipelines, especially in settings involving thousands of judgments per experimental batch.

```
def label_rows(df):
    completed = 0
    batch = 0

    # This is the trick to parallelize the code across rows and allow for "resuming" from a checkpoint
    results = [None] * len(df)
    tasks = [
        (i, row["instruction"], row["answer"])
        for i, (_, row) in enumerate(df.iterrows())
    ]

    # Parallelize across rows
    with ThreadPoolExecutor() as executor:
        for i, scores in executor.map(eval_row, tasks):
            results[i] = scores
            completed += 1

            # This is a really kinky way to save progress but it worked :p
            if completed % 100 == 0:
                print(f"Completed {completed} rows! Saving progress...")

                # The so-called "progress" files
                with open(f'progress_{batch}.pkl', 'wb') as f:
                    pickle.dump(results, f)

                batch += 1
```

Figure 4: Parallelization of the scoring of rows, where every operation is also parallelized using the exponential back-off approach. This allows for fast rubric judging with the Martian API across big datasets.