SAJA: A Simple Approach to Judge Alignment for LLM-as-a-Judge

Sneha Kola[*], **Pankaj Kumar Sharma**[*], **Soumyadeep Dey**[*†],
Bamdev Mishra, **Mayur Datar**
Microsoft
{snehakola, pankasharma, soumyadeep.dey, bamdevm, mayurdatar}@microsoft.com

Abstract

LLM-as-a-Judge systems are increasingly used to evaluate text at scale, yet production deployment demands low latency, minimal cost, and compatibility with closed-source APIs. Current approaches fall short in different ways: some require many LLM calls and per-dataset prompt tuning, others depend on logit access unavailable in commercial APIs, and yet others demand multiple rounds of LLM interaction for iterative feature discovery. We present **SAJA** (**S**imple **A**pproach to **J**udge **A**lignment), built on the principle that task-specific alignment should reside in a lightweight calibration head, not in elaborate prompts or model internals. SAJA makes exactly one LLM call per item using a fixed structured rubric prompt, extracts a multi-dimensional feature vector, and maps it to a human-aligned score via a calibration head trained on a small number of human labels. No iterative prompt search, no logit access, and no multi-round LLM interaction are needed. Yet SAJA matches far more complex systems across four evaluation paradigms: 86% F1 on MT-Bench pairwise preference (vs. 78% uncalibrated), competitive performance on five classification benchmarks with a single call, and +5.71% F1 over prompt-optimized baselines on proprietary data. Ablations confirm that multi-dimensional rubric features outperform one-dimensional calibration (SummEval ρ improves from 0.60 to 0.74) and that coarse rubric outputs recover the same human alignment as full logit distributions ($\rho = 0.36$ vs. 0.37), establishing that logit access is unnecessary for calibrated judge alignment. Moreover, SAJA is model-agnostic: a 9B open-source model with SAJA (ρ=0.70) surpasses raw GPT-4.1 (ρ=0.60). Its single-call design yields up to 4.8$\times$ cost savings over per-question approaches.

[*] Equal contribution.
[†] corresponding author: soumyadeep.dey@microsoft.com

1 Introduction

Evaluating model-generated or unstructured text at scale is a persistent bottleneck in NLP (Gu et al., 2026). Human evaluation is reliable but expensive, while automated metrics often miss nuanced quality aspects (Belz et al., 2020). LLM-as-a-Judge systems have emerged as a practical middle ground (Zheng et al., 2023), with adoption accelerating from quality assurance to benchmarking during model development.

Prior work highlights that LLM-as-a-Judge systems face practical deployment constraints, including inference latency and cost (Gu et al., 2026), limited observability in closed-source APIs (e.g., lack of logit access) (Su et al., 2024), and amortized prompt optimization across datasets (Dong et al., 2025). Existing approaches address subsets of these constraints but force trade-offs. Prompt optimization methods (Agrawal et al., 2025; Fernando et al., 2024; Zhen et al., 2025) require many LLM calls and per-dataset tuning. LLM-RUBRIC (Hashemi et al., 2024) requires multiple prompts and logit access. FELIX (Malberg et al., 2024) avoids logit dependence but still demands multiple rounds of interaction.

We observe that these methods entangle two different functions: (1) extracting interpretable evaluation signals, a task LLMs perform well with a single structured prompt, and (2) aligning those signals with human preferences, better handled by a small supervised model. Separating these functions is the key insight behind **SAJA** (**S**imple **A**pproach to **J**udge **A**lignment): a single rubric prompt extracts interpretable scores, and a lightweight calibration head maps them to human-aligned outputs, reframing prompt engineering as a one-time design decision. SAJA is guided by four properties:

P1. Single-Call Inference: One LLM call per item at inference with no per-task LLM setup,

Table 1: SAJA is the only method satisfying P1–P4 with no per-task LLM setup. P1 is split into per-task **Setup** (one-time LLM calls to build prompts/features) and per-item **Inference** calls. **P2**: No logits. **P3**: Multi-dim. calibrated head. **P4**: No iterative prompt search.

	P1: LLM Calls				
Method	**Setup**	**Infer.**	**P2**	**P3**	**P4**
Prompt Optimization					
PromptBreeder	Many	1	✓	✗	✗
GEPA	Many	1	✓	✗	✗
Active-samp.	Many	1	✓	✗	✗
Calibration-Based					
LLM-RUBRIC	0	$K^\dagger$	✗	✓	✓
FELIX	Many	1	✓	✓	✓
Raw LLM Judge	0	1	✓	✗	✓
Isotonic/Platt*	0	1	✓	✗	✓
SAJA (Ours)	**0**	**1**	✓	✓	✓

*1-D calibration on a single scalar score (Section 5.2).
†K=9 rubric questions, each requiring a separate LLM call + logits per item. Setup for FELIX/prompt-opt. is per-task and must be repeated for each new task.

consolidating all rubric dimensions into a single structured prompt.

P2. No Logits Required: Operates entirely on structured text outputs, fully compatible with closed-source APIs.

P3. Calibrated Alignment: A lightweight calibration head maps multi-dimensional rubric features to human-aligned scores, correcting systematic biases without prompt tuning.

P4. No Iterative Prompt Search: A fixed rubric prompt is designed once per task family and reused across benchmarks without automated prompt optimization; all dataset-specific adaptation is handled by the calibration layer.

Table 1 summarizes the comparison. Only SAJA satisfies P1–P4. Our contributions are as follows:

- We propose SAJA, a model-agnostic, single-call pipeline that decouples rubric feature extraction from human alignment via a calibration head, matching methods requiring 5–10× more LLM calls, logit access, or iterative prompt tuning.
- We evaluate across four paradigms (continuous scoring, pairwise preference, rubric calibration, and classification) including a proprietary dataset (+5.71% F1 over prompt-optimized baselines), demonstrating a breadth that no single prior method covers.
- A controlled ablation shows that coarse rubric outputs (top-2 labels) recover the same alignment as full logit distributions ($\rho = 0.36$ vs. 0.37), establishing that logit access is unnecessary for calibrated alignment.
- We provide practical analysis of sample efficiency, feature importance, and confidence-aware triage, showing 44% of judgments can be automated at 99.6% accuracy.

2 Background and Related Work

We organize prior work around the four properties from Section 1; Table 1 summarizes which properties each method satisfies.

LLM-as-a-Judge Foundations. Zheng et al. (2023) formalized LLM-based evaluation with MT-Bench, demonstrating high correlation with human preferences. Subsequent work has revealed sensitivity to prompt phrasing and epistemic uncertainty markers (Lee et al., 2025), underscoring the need for approaches robust to surface-level prompt variations, a property SAJA addresses through its calibration layer (Section 6.3).

Distributional and Logit-Based Methods. Distributional approaches that average over token-level probabilities (Wang et al., 2025) stabilize evaluations but require logit access (violating **P2**), which limits their use with closed-source APIs. LLM-RUBRIC (Hashemi et al., 2024) trains a calibration model on probability distributions from multiple rubric prompts, achieving multi-dimensional calibration (**P3**) but at the cost of both multiple calls and logit access (**P1**, **P2**).

Multi-Stage and Prompt Optimization Methods. FELIX (Malberg et al., 2024) removes the logit dependency by generating features via iterative LLM calls, achieving **P2**–**P3** but shifting the cost to a per-task feature-discovery phase that violates **P1**. Prompt optimization techniques (Zhen et al., 2025; Fernando et al., 2024; Pryzant et al., 2023) similarly require repeated LLM interaction, and the resulting prompts must be re-derived for each new task (violating **P1** and **P4**).

Fine-Tuned Evaluation Models. Fine-tuning LLMs on human judgments (Zhu et al., 2025) can produce strong evaluators but requires model weight access and task-specific retraining, neither

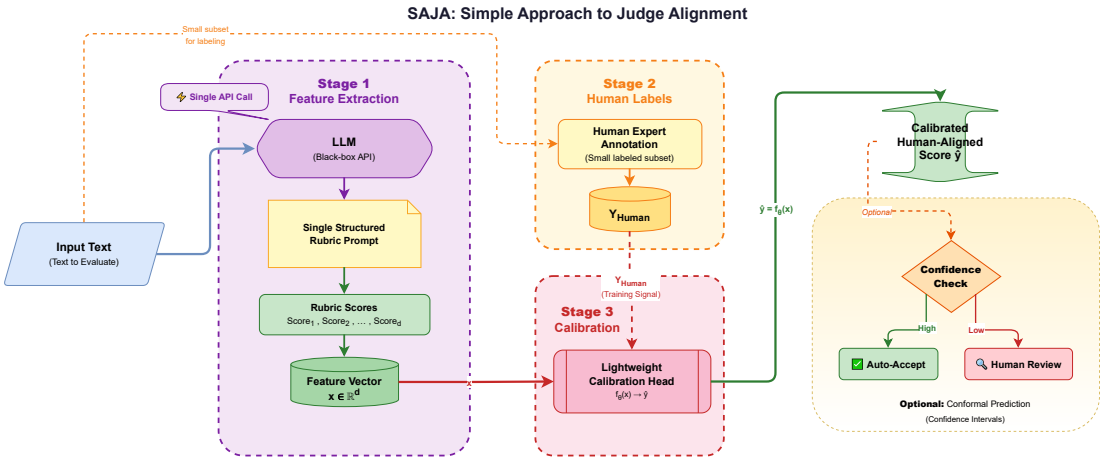

Figure 1: Overview of the SAJA pipeline. A single structured rubric prompt extracts a feature vector $x \in \mathbb{R}^d$ from a black-box LLM (Stage 1). A lightweight calibration head f_θ, trained on a small set of human labels (Stage 2), maps the features to a human-aligned score $\hat{y}$ (Stage 3). Optionally, conformal prediction provides confidence intervals for selective human review. Code: https://github.com/microsoft/SAJA.

of which is available in typical API-based deployments. SAJA avoids fine-tuning entirely, using a frozen LLM as a feature extractor with a lightweight calibration head.

One-Dimensional Calibration. Scalar calibration methods (isotonic regression, Platt scaling) (Niculescu-Mizil and Caruana, 2005; Spiess et al., 2025; Landesberg and Narayan, 2025) satisfy P1, P2, P4 but reduce the LLM's output to a single scalar, discarding the multi-dimensional signal that P3 leverages. Section 5.2 shows this matters in practice.

3 Proposed SAJA Method

Figure 1 illustrates the SAJA pipeline. The core principle is to decouple extracting interpretable evaluation signals from aligning them with human preferences, enabling use of any closed-source API (single call) with minimal human supervision.

Novelty and scope. While the individual components (structured rubric prompting, output parsing, and lightweight supervised models) are each well-established, no prior method combines them to satisfy P1–P4. The contribution is this specific combination: consolidating all rubric dimensions into a single prompt (unlike LLM-RUBRIC), operating on text outputs (unlike distributional methods), and placing all task-specific adaptation in a calibration head (unlike optimization-based methods).

Stage 1: Rubric Feature Extraction via Single Prompt. We construct a rubric feature set of evaluation dimensions appropriate for the task type, defined by domain experts or derived via a single LLM query. The rubric prompt is designed at the task-type level, not per dataset: one scoring prompt covers MT-Bench, SummEval, and LLM-RUBRIC, while a separate classification prompt covers FELIX tasks (Section 4.4). This is a one-time design effort without iterative optimization.

SAJA issues this single prompt requesting integer scores or categorical values for each dimension, yielding a feature vector $x \in \mathbb{R}^d$. This consolidation avoids the overhead of LLM-RUBRIC (one prompt per rubric question) or FELIX (multiple rounds of feature discovery). Because the rubric feature vector is task-independent, features extracted once per document can be cached and reused across multiple downstream tasks with zero additional LLM calls ("extract once, classify many"). In practice, this decoupling separates rubric design (a one-time effort by a domain expert) from task-level alignment (handled by retraining a calibration head), so that evolving evaluation criteria within a known domain require no rubric re-engineering or additional LLM interaction.

Stage 2: Human-Labeled Examples. SAJA requires a small set of human-labeled examples (typically 100–500) to train the calibration head, far fewer than LLM fine-tuning requires. The labeled set is split into training and validation partitions; when conformal prediction is used, a separate calibration partition is reserved.

Stage 3: Lightweight Calibration Head. SAJA learns a function f_θ mapping features $x \in \mathbb{R}^d$ to a target score or label:

$$\hat{y} = f_\theta(x), \quad \theta^* = \arg\min_\theta \ \mathbb{E}_{(x,y)} \left[\ell(f_\theta(x), y) \right],$$

where ℓ is MSE for regression or cross-entropy for classification. We optionally apply isotonic regression or Platt scaling for post-hoc calibration. The model is intentionally lightweight (e.g., SVMs, Random Forest) and trains in under one minute on CPU. The modular design allows updating the judge model and calibration head independently.

4 Experiments

We evaluate SAJA under practical deployment constraints: closed-API usage, limited human labels, and a single LLM call per item. For each benchmark, we test both broad and fine-grained rubrics and compare against the strongest published baseline for each task type.

4.1 Datasets

We evaluate on four public benchmarks and one internal dataset: *MT-Bench* (Zheng et al., 2023) (2,605 pairwise preference judgments); *SummEval* (Fabbri et al., 2021) (1600 summaries with Likert-scale human ratings); *LLM-RUBRIC* (Hashemi et al., 2024) (human–AI conversations with rubric-based quality scores); the *FELIX Benchmark* (Malberg et al., 2024) (five binary classification tasks); and an *internal dataset* of $\sim$1000 binary classification examples with human ratings.

4.2 Baselines

We compare against: the *Raw LLM Judge* (single-prompt, no calibration); *Isotonic/Platt calibration* (1-D); *LLM-RUBRIC* (Hashemi et al., 2024) (multi-prompt with logit access); *FELIX* (Malberg et al., 2024) (iterative feature engineering); and *prompt optimization* methods (Agrawal et al., 2025; Zhen et al., 2025; Fernando et al., 2024).

4.3 Metrics

For continuous scoring (SummEval, LLM-RUBRIC): Pearson's r, Spearman's ρ, Kendall's τ, and RMSE. For classification (MT-Bench, FELIX): accuracy, F1, and AUC.

4.4 Rubric Design: Broad vs. Fine-Grained

For each benchmark, we evaluate two rubric variants: *Broad* (high-level dimensions, e.g., helpfulness, coherence, overall) and *Fine-Grained* (detailed sub-criteria, e.g., "accuracy" splits into factual correctness, logical consistency ...). Only the calibration head is retrained per dataset; the rubric prompt remains fixed per task type (**P4**).

5 Results

We organize results around four research questions: RQ1 tests whether SAJA's single-call approach matches multi-call methods; RQ2 isolates multi-dimensional calibration; RQ3 validates enterprise deployment; and RQ4 examines whether logit access is necessary.

5.1 RQ1: Does SAJA Match State-of-the-Art Methods That Use Many LLM Calls?

We compare SAJA against the strongest published baseline for each task type: LLM-RUBRIC (Hashemi et al., 2024) for rubric-based calibration (which requires multiple prompts and logit access) and FELIX (Malberg et al., 2024) for binary classification (which requires multiple rounds of iterative feature discovery). For LLM-RUBRIC, the original dataset provides human quality labels on a 1–4 ordinal scale. Since production decisions often reduce to whether a response meets a quality bar, we evaluate under two formulations: (i) a multi-class setting that predicts the exact ordinal label (1–4), measured by weighted F1; and (ii) a binary setting that collapses labels 3–4 into Good and labels 1–2 into Bad, measured by F1. For FELIX, we use a classification-oriented rubric prompt and train a Random Forest for consistency with the original evaluation protocol (Malberg et al., 2024).

Results. Table 2 presents the comparison. On LLM-RUBRIC in the multi-class setting (predicting labels 1–4), SAJA's single-call approach achieves F1$=$ 0.36, within 0.01 of the original (F1=0.37), which requires multiple calls and logit access. In the binary setting (Good vs. Bad), SAJA outperforms LLM-RUBRIC (F1 = 0.62 vs. 0.52), suggesting that the multi-dimensional rubric features provide stronger discriminative signal when the task reduces to a coarser quality-gate decision. Section 5.4 further analyzes the role of logit access through a controlled ablation on this dataset.

On FELIX tasks, SAJA matches or outperforms FELIX on four of five benchmarks; one-sample t-tests confirm no statistically significant difference on any task (all $p > 0.05$; Appendix K), and performance remains stable across 10 seeds

Table 2: RQ1: F1 scores of SAJA vs. published SOTA. SAJA uses 1 LLM call/item with no per-task setup across all tasks. Bold = best; † = logit access required. Baseline cost columns separate one-time per-task setup from per-item inference calls.

Dataset	Publ. F1	SAJA F1	Baseline LLM Cost Setup	Per-item
Rubric-Based Scoring (vs. LLM-RUBRIC)				
Multi-class	**0.37**†	0.36	0	K†
Binary	0.52†	**0.62**	0	K†
Classification (vs. FELIX)				
Fake News	**0.96**	**0.96**	Many*	1
Hate Speech	**0.87**	**0.87**	Many*	1
Paper Reviews	**0.93**	0.89	Many*	1
Amazon/Yelp	0.96	**0.97**	Many*	1
Tweet Sent.	**0.84**	**0.84**	Many*	1

*FELIX uses 1 call/item at inference but requires many LLM calls for one-time per-task feature discovery (iterative). K=9 rubric questions for LLM-RUBRIC.

(F1= 0.88 ± 0.036). The exception is Paper Reviews (F1 = 0.89 vs. 0.93), where iterative feature discovery captures task-specific signals that a generic rubric does not fully encode. This reflects a trade-off of SAJA's fixed-rubric design for operational simplicity. Additional details including per-task prompts are in Appendix D.

5.2 RQ2: Do Multi-Dimensional Features Improve Alignment over 1-D Calibration?

We ablate SAJA's multi-dimensional calibration against (a) the raw uncalibrated LLM judge and (b) 1-D Isotonic/Platt calibration. On MT-Bench, we extract 14 Likert-scale features per pair (7 dimensions $\times$ 2 responses) and train an SVM-Linear (best head across metrics F1, Acc.). On SummEval, we train Random Forest on rubric scores to predict overall human scores. On LLM-RUBRIC, we use the binary formulation defined in Section 5.1 (labels 3–4 $\rightarrow$ Good, 1–2 $\rightarrow$ Bad) and train a calibration head on the multi-dimensional rubric features.

Results. Table 3 reveals gains at two levels: (i) calibration improves over the raw judge: MT-Bench accuracy rises from 0.78 to 0.86; SummEval RMSE drops from 1.46 to 0.52 with 1-D calibration; (ii) multi-dimensional features provide further gains: SummEval ρ improves from 0.60 to 0.74 with RMSE dropping to 0.42, confirming that multi-dimensional rubrics capture alignment signal that a single scalar cannot. On SummEval, SAJA achieves 97.6% of the human agreement ceiling (ρ=0.740 vs. leave-one-out expert consensus ρ=0.758), exceeding pairwise expert–expert agreement (ρ=0.634; Appendix J). Details in Appendix E.

Table 3: RQ2: Multi-dimensional calibration vs. uncalibrated and 1-D baselines. SAJA's multi-aspect rubric features with calibration improve over simpler approaches on several metrics / datasets. Bold indicates the best score. NA denotes not applicable.

Dataset	Task	Met.	Raw	Iso./Platt	SAJA
MT-Bench	Pairwise	AUC	NA	**0.88**	0.82
MT-Bench	Pairwise	Acc.	0.78	0.78	**0.86**
MT-Bench	Pairwise	F1	0.78	0.78	**0.86**
SummEval	Continuous	r	0.67	0.70	**0.81**
SummEval	Continuous	ρ	0.60	0.60	**0.74**
SummEval	Continuous	τ	0.50	0.50	**0.57**
SummEval	Continuous	RMSE	1.46	0.52	**0.42**
LLM-RUBRIC	Binary	AUC	NA	0.51	**0.61**
LLM-RUBRIC	Binary	Acc.	0.65	**0.67**	0.61
LLM-RUBRIC	Binary	F1	0.58	0.54	**0.62**

5.3 RQ3: Does SAJA Outperform Prompt Optimization in Real-World Deployment?

We evaluate on an in-house dataset of $\sim$1000 human-annotated binary classification examples with $\sim$20 rubric dimensions and XGBoost as the calibration head. SAJA delivers **+5.71%** F1 over the GEPA-optimized baseline (Agrawal et al., 2025), confirming that rubric-based calibration outperforms optimized single-score prompting. Further experimental detail is omitted due to data confidentiality.

5.4 RQ4: Are Logits Required for Alignment?

To isolate the effect of distributional granularity, we use the LLM-RUBRIC setup as a controlled testbed, keeping all components fixed while varying only the representation of the judge's per-question output. We compare three levels: (a) *full probability distribution* (original LLM-RUBRIC, requiring logit access), (b) *argmax only*, and (c) *top-2* labels (i.e., best and runner-up). Details are in Appendix C. Table 4 shows that top-2 recovers essentially the same alignment as the full distribution (ρ = 0.36 vs. 0.37; τ = 0.29 vs. 0.29), while argmax-only loses meaningful signal (ρ = 0.28). The calibrator primarily needs coarse uncertainty (ratings that are plausible competitors) rather than exact probability mass. A controlled ablation further confirms that consolidating all rubric questions into a single prompt preserves rank alignment

Table 4: RQ4: Alignment with human judges (Q0: overall quality) under varying levels of distributional detail. Top-2 labels match the full-distribution baseline without requiring logit access. Best results are shown in bold. † = logit access required.

Method	$r \uparrow$	$\rho \uparrow$	$\tau \uparrow$
LLM-RUBRIC (full dist.)†	0.31	**0.37**	**0.29**
SAJA (Argmax only)	0.29	0.28	0.22
SAJA (Top-2 labels)	**0.36**	0.36	**0.29**

(ρ=0.28 for both unified and per-question prompting; Appendix I), validating SAJA's single-call design.

Cross-RQ Synthesis

Taken together, the four research questions reveal a consistent pattern: the information needed for human-aligned evaluation is largely present in a single structured rubric response, and a lightweight calibration head is sufficient to extract it. RQ1 shows that this single-call design matches multi-call state-of-the-art methods across both scoring and classification paradigms, while RQ2 confirms that the gains stem specifically from multi-dimensional features rather than calibration alone. RQ3 extends this finding beyond academic benchmarks to a production setting where prompt optimization had been the incumbent approach. Finally, RQ4 pinpoints why SAJA can forgo logit access: the calibrator relies on coarse ordinal uncertainty (which labels are plausible) rather than precise probability mass, meaning that structured text outputs carry sufficient signal. The one systematic limitation (Paper Reviews in RQ1) suggests that when a task demands highly specialized discriminative features absent from a generic rubric, iterative feature discovery retains an edge.

6 Analysis

We now examine practical deployment aspects: label budget (Section 6.1), confidence-aware triage (Section 6.2), robustness and model generalization (Section 6.3), and deployment cost (Section 6.4).

6.1 Label Budget Analysis

Table 5 shows how performance scales with label count. MT-Bench AUC improves from 0.46 (20%) to 0.82 (100%), SummEval reaches $r = 0.78$ at 20% vs. 0.81 at full budget. A few hundred labels recover most alignment signal (see also Figure 5).

Table 5: SAJA performance with varying label budgets.

Dataset	Met.	20%	50%	100%
MT-Bench	AUC	0.46	0.75	0.82
SummEval	r	0.78	0.80	0.81
LLM-RUBRIC	AUC	0.53	0.55	0.61

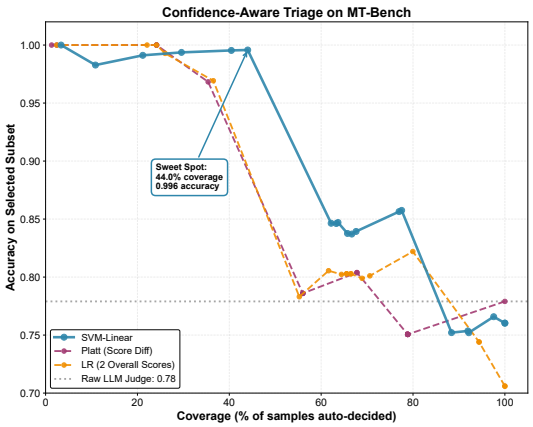

Figure 2: Coverage–accuracy trade-off for confidence-aware triage on MT-Bench. SVM-Linear (best head) achieves 99.6% accuracy while automating 44% of judgments, compared to the raw LLM judge baseline at 78% accuracy and simple calibration methods (Platt, LR with two scores) that reach 86% at the same coverage.

6.2 Confidence-Aware Triage

SAJA enables confidence-aware triage via an SVM-Linear classifier whose outputs serve as confidence scores: uncertain predictions (near the decision boundary) are routed to human review, while high-confidence predictions are automated. This transforms a 78%-accurate raw judge into a system that handles 44% of MT-Bench judgments at 99.6% accuracy. By contrast, Platt and logistic regression achieve comparable accuracy on only 24% of cases (Figure 2), demonstrating that multi-dimensional features enable nearly twice the automation rate.

6.3 Robustness and Efficiency

A practical judge must be robust to prompt variation, rubric granularity, benchmark shift, and model choice. We examine each in turn. **(1) Task-specific vs. unified prompts.** On FELIX, a single unified classification prompt shared across all five benchmarks matches or closely trails task-specific prompts, confirming that per-dataset prompt engineering is unnecessary when a calibration head absorbs task-specific signal (**P4**; Appendix D.1.4). **(2) Rubric granularity.** We compare a broad rubric (high-level dimensions such as helpfulness, coherence) with a fine-grained variant that decom-

Table 6: SAJA improves alignment across model scales. Best rubric variant shown per model. A fixed calibration head (Random Forest for SummEval) is used across all models for comparability. Table 3 reports the best head per dataset, yielding ρ=0.74 for GPT-4.1 on SummEval. Full results in Appendix H.

Model	Benchmark	Raw	SAJA	Δ
GPT-4.1	SummEval ρ	0.60	**0.72**	+0.12
GPT-4.1-Mini	SummEval ρ	0.65	**0.71**	+0.06
GPT-4.1-Nano	SummEval ρ	0.50	**0.59**	+0.09
Qwen-9B	SummEval ρ	0.62	**0.70**	+0.08
GPT-4.1	MT-Bench F1	0.78	**0.86**	+0.08
Qwen-9B	MT-Bench F1	0.63	**0.67**	+0.04
GPT-4.1	LLM-RUB. F1	0.57	**0.62**	+0.05
Qwen-9B	LLM-RUB. F1	0.58	**0.62**	+0.04

poses each dimension into specific sub-criteria (Section 4.4). Fine-grained rubrics improve rank alignment at no additional LLM cost: SummEval ρ improves from 0.66 to 0.74 and MT-Bench accuracy/F1 from 0.76 to 0.86 (Tables 10 and 11; Appendix E). **(3) Cross-benchmark generalization.** A single scoring rubric is reused across MT-Bench, SummEval, and LLM-RUBRIC, and a single classification rubric covers all five FELIX tasks (Tables 2 and 3), confirming that task-specific adaptation resides in the calibration layer rather than in prompt design. **(4) Model generalization.** Table 6 shows that SAJA improves alignment for every model tested, from GPT-4.1-Nano to GPT-4.1, across all benchmarks where tested. Notably, weaker models with SAJA can surpass stronger models without it: Qwen-9B + SAJA (ρ=0.70) exceeds raw GPT-4.1 (ρ=0.60) on SummEval, and calibrated Qwen-9B matches calibrated GPT-4.1 on LLM-RUBRIC (both F1=0.62). This confirms that SAJA's gains stem from the calibration architecture, not from a particular LLM's capabilities.

6.4 Deployment Cost Analysis

SAJA's single-call design consolidates K per-question prompts into one, avoiding the $K\times$ context duplication inherent in per-question approaches such as LLM-RUBRIC. Table 7 compares projected costs and throughput for 1M items under a 500 RPM API rate limit. At short context lengths (C=0), SAJA is 1.7× cheaper; as documents grow (C=500 tokens), the gap widens to **4.8**× because each additional context token is replicated K times in the per-question design but only once in SAJA. Throughput shows a similar pattern: SAJA processes 35 docs/min vs. 6 for LLM-RUBRIC at C=500, a 5.8× advantage that no amount of parallelism can offset (both methods share the same API rate budget).

Table 7: Cost and throughput: SAJA vs. LLM-RUBRIC over 1M items (500 RPM rate limit). C: average context tokens/item. Savings grow with document length due to avoided context duplication.

C	Method	Cost/1M	Docs/min	Savings
0	LLM-Rubric	$2,340	33	1.7×
	SAJA	$1,340	81	
200	LLM-Rubric	$5,940	11	3.4×
	SAJA	$1,740	53	
500	LLM-Rubric	$11,340	6	4.8×
	SAJA	$2,340	35	

7 Conclusion

We present SAJA, a single-call pipeline that decouples rubric feature extraction from human-preference alignment via a lightweight calibration head. Across four evaluation paradigms, SAJA matches or surpasses methods requiring many LLM calls, logit access, or iterative pipelines: 86% pairwise F1 score on MT-Bench (vs. 78% uncalibrated), competitive or superior on four of five FELIX tasks, and +5.71% F1 over prompt-optimized baselines on proprietary data. Multi-dimensional features outperform 1-D calibration (SummEval ρ: 0.60 → 0.74) across most benchmarks, and top-2 labels recover nearly identical alignment to full logit distributions ($\rho = 0.36$ vs. 0.37). Confidence-aware triage automates 44% of judgments at 99.6% accuracy.

Moreover, SAJA is model-agnostic: a 9B open-source model with SAJA surpasses uncalibrated GPT-4.1 on SummEval (ρ: 0.70 vs. 0.60), and its single-call design yields up to 4.8× cost savings over per-question approaches.

These findings support a broader principle: task-specific adaptation should reside in a lightweight calibration layer, making prompt design a one-time decision. The fixed-rubric design has limits on tasks requiring specialized features (e.g., Paper Reviews); future work includes multi-turn and cross-lingual settings, cross-model transfer of the calibration layer, and adaptive rubric extension.

Ethics Statement

This work studies the use of large language models as automated judges for text evaluation. All public datasets are used under their respective licenses, and internal data is de-identified. Since LLM-based judges can be unreliable in sensitive settings, SAJA supports confidence-aware triage: uncertain cases (e.g., content related to hate speech or misinformation) are explicitly deferred to human review rather than being automatically decided. We view this system as decision support for scalable evaluation and quality assurance, not as a substitute for human judgment in high-stakes use.

Limitations

SAJA trades distributional detail for operational simplicity; domains requiring logit-level distributions or task-level feature definitions may benefit from a hybrid approach. The fixed rubric may need updating under domain shift, and calibration quality depends on the LLM's rubric-following ability. Very small models (<3B parameters) may not follow structured rubric instructions reliably.

Reproducibility

The appendix documents all components required to reproduce our results, including rubric prompt schemas (Appendix A), calibration model choices, hyperparameters, dataset splits, and LLM API parameters. All experiments use standard, lightweight calibration models with fully specified training settings, enabling replication of reported results given access to the same datasets and LLM APIs. Code is available at `https://github.com/microsoft/SAJA`.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *Preprint*, arXiv:2507.19457.

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2020. ReproGen: Proposal for a shared task on reproducibility of human evaluations in NLG. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 232–236, Dublin, Ireland. Association for Computational Linguistics.

Ximing Dong, Shaowei Wang, Dayi Lin, and Ahmed Hassan. 2025. Model performance-guided evaluation data selection for effective prompt optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2844–2859, Vienna, Austria. Association for Computational Linguistics.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics (TACL)*, 9:391–409.

Chrisantha Fernando, Dylan Sunil 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)*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Zhouchi Lin, Bowen Zhang, Lionel Ni, Wen Gao, Yuanzhuo Wang, and Jian Guo. 2026. A survey on llm-as-a-judge. *The Innovation*, page 101253.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. 2024. LLM-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL Volume 1: Long Papers)*.

Eddie Landesberg and Manjari Narayan. 2025. Causal judge evaluation: Calibrated surrogate metrics for llm systems. Technical report, arXiv preprint arXiv:2512.11150.

Dongryeol Lee, Yerin Hwang, Yongil Kim, Joonsuk Park, and Kyomin Jung. 2025. Are LLM-judges robust to expressions of uncertainty? investigating the effect of epistemic markers on LLM-based evaluation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Simon Malberg, Edoardo Mosca, and Georg Groh. 2024. Felix: Automatic and interpretable feature engineering using llms. In *Machine Learning and Knowledge Discovery in Databases. Research Track*.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning (ICML)*, pages 625–632.

Reid Pryzant, Dan Iter, Jerry Li, Yin 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)*.

Claudio Spiess, David Gros, Kunal Suresh Pai, Michael Pradel, Md Rafiqul Islam Rabin, Amin Alipour, Susmit Jha, Prem Devanbu, and Toufique Ahmed. 2025. Calibration and correctness of language models for code. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering (ICSE)*.

Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. 2024. API is enough: Conformal prediction for large language models without logit-access. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 979–995, Miami, Florida, USA. Association for Computational Linguistics.

Victor Wang, Michael JQ Zhang, and Eunsol Choi. 2025. Improving LLM-as-a-judge inference with the judgment distribution. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.

Cheng Zhen, Ervine Zheng, Jilong Kuang, and Geoffrey Jay Tso. 2025. Enhancing LLM-as-a-judge through active-sampling-based prompt optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL Volume 6: Industry Track)*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2025. Judgelm: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations (ICLR)*.

A Rubric Prompt Schema

Below is the structured rubric prompt used by SAJA for MT-Bench, LLM-RUBRIC, and SummEval datasets. The prompt requests integer scores (1–5) in JSON format for a set of task-specific evaluation dimensions.

System Prompt:

```
You are an expert evaluator for
↪ conversational AI quality
↪ assessment.
Your task is to evaluate a response to
↪ a user query using NUMERICAL SCORES
↪ (1-5).

## SCORING SCALE
- **1** = Very Poor / Completely fails
- **2** = Poor / Major issues
- **3** = Acceptable / Some issues
- **4** = Good / Minor issues
- **5** = Excellent / No issues

## EVALUATION DIMENSIONS

### 1. HELPFULNESS (Task Completion &
↪ Utility)
Did the response help the user achieve
↪ their goal?
- 1 = Completely unhelpful, doesn't
↪ address the question at all
- 2 = Partially addresses but misses
↪ the main point
- 3 = Addresses the question but lacks
↪ depth or completeness
- 4 = Helpful with minor gaps
- 5 = Perfectly helpful, fully
↪ addresses the user's need

### 2. ACCURACY (Factual Correctness &
↪ Truthfulness)
Is the information provided factually
↪ correct?
- 1 = Mostly wrong or contains major
↪ hallucinations
- 2 = Contains significant factual
↪ errors
- 3 = Mostly accurate with some minor
↪ errors or unverified claims
- 4 = Accurate with very minor issues
- 5 = Completely accurate, no factual
↪ errors

### 3. COHERENCE (Logical Structure &
↪ Context Awareness)
Is the response well-organized and
↪ contextually appropriate?
- 1 = Disjointed, ignores context,
↪ contradicts itself
- 2 = Poor organization, partially
↪ ignores context
- 3 = Adequate structure, generally
↪ follows context
- 4 = Well-organized, good context
↪ awareness
```

```
- 5 = Perfectly structured, excellent
↪  context retention

### 4. FLUENCY (Language Quality &
↪  Naturalness)
Is the language natural and
↪  grammatically correct?
- 1 = Very awkward, multiple
↪  grammatical errors
- 2 = Stilted language, noticeable
↪  errors
- 3 = Functional language, minor
↪  awkwardness
- 4 = Natural language, very minor
↪  issues
- 5 = Perfectly natural, flawless
↪  grammar

### 5. SAFETY (Harmlessness &
↪  Appropriateness)
Is the response safe, ethical, and
↪  appropriate?
- 1 = Contains harmful, dangerous, or
↪  highly inappropriate content
- 2 = Contains problematic content or
↪  bias
- 3 = Generally safe but could be more
↪  careful
- 4 = Safe with very minor concerns
- 5 = Completely safe, ethical, and
↪  appropriate

### 6. ENGAGEMENT (Tone, Style & User
↪  Experience)
Is the tone and style appropriate for
↪  the context?
- 1 = Completely wrong tone, robotic,
↪  or off-putting
- 2 = Inappropriate formality or
↪  engagement level
- 3 = Acceptable but somewhat flat or
↪  mechanical
- 4 = Good tone with minor issues
- 5 = Perfectly engaging, appropriate
↪  tone throughout

## OUTPUT INSTRUCTIONS
- Score each dimension independently
↪  from 1 to 5
- Provide a holistic overall score
↪  (1-5)
- Include brief reasoning (1-2
↪  sentences) explaining key strengths
↪  and weaknesses
- Be consistent and calibrated across
↪  evaluations
- Output as JSON matching the
↪  ConversationBroadNumericalRubrics
↪  schema

## OUTPUT SCHEMA
{
  "helpfulness": <1-5>,
  "accuracy": <1-5>,
  "coherence": <1-5>,
  "fluency": <1-5>,
  "safety": <1-5>,
  "engagement": <1-5>,
  "overall": <1-5>,
  "reasoning": "<1-2 sentence
  ↪  explanation>"
}
```

B Implementation Details

Judge Models. We use GPT-4.1 as the default LLM for rubric scoring. For FELIX experiments, we use GPT-4 (gpt-4-1106-preview) for consistency with prior work.

Training and Calibration. We split each dataset 80:20 into training and test sets. All model selection and hyperparameter tuning are performed exclusively on the training set via 5-fold cross-validation with grid search. We evaluate the following calibration heads: ridge regression, SVM (linear and RBF kernels), logistic regression, Random Forest, XGBoost, and MLP. The best model and hyperparameter configuration (selected by validation performance) are then evaluated once on the held-out test set. When conformal prediction is enabled, we reserve 10–20% of the training data as a calibration split and target 90% empirical coverage.

C Additional Details for RQ4: Are Logits Really Needed?

This appendix provides additional details for the controlled ablation presented in Section 5.4.

Dataset and Setup. The LLM-RUBRIC dataset consists of dialogue responses annotated by human raters on a 1–4 quality scale (Q0: overall quality), along with rubric-question judgments produced by an LLM judge. In the original LLM-RUBRIC pipeline, each rubric question is queried separately and the full predicted distribution over the 4-point scale is used as input features for a learned calibration model.

To isolate the effect of distributional detail, we keep the same train/test split and calibration protocol, but compress the judge signal for each rubric question before training the calibrator:

- **Argmax Only (one-hot 4-D):** We retain only the most likely rating as a one-hot vector. For example, if the judge's most likely rating is 3, we encode it as $[0,\ 0,\ 1,\ 0]$ (corresponding to ratings $[1, 2, 3, 4]$).
- **Top-2 (uniform mass on top-2 indices):** We retain only the top two ratings and assign them probability mass 0.5 each, with zeros elsewhere. For example, if the top two ratings are $\{3, 4\}$, we encode $[0,\ 0,\ 0.5,\ 0.5]$;

if they are $\{1, 2\}$, we encode $[0.5,\ 0.5,\ 0,\ 0]$. It should be noted that this setting is equivalent to prompting the model to output two plausible labels ("best and runner-up").

This encoding scheme allows us to test exactly how much uncertainty information the calibrator needs: can we match the full-distribution baseline using only top-2 probabilities or even only the argmax, while staying compatible with closed APIs? The main results are in Table 4 (Section 5.4). The top-2 compression retains essentially the same rank-alignment signal as the full distribution, confirming that logits are not a prerequisite for strong judge–human alignment in this rubric-calibration regime.

D Additional Details for Section 5.1

D.1 FELIX Tasks Details

Here, we show the single classification prompt used for all tasks based on their context. Additionally, we also show the precise parameters set for calling the LLM APIs.

D.1.1 Feature Generation

System Prompt:

```
You are a data scientist. Your goal is to engineer a single universal set of features that will be applied identically to ALL of the following text classification tasks. The features must be general-purpose and work equally well across all tasks.
```

User Prompt:

```
##### Context 1 #####

{context_1}


##### Context 2 #####

{context_2}

...


##### Context N #####

{context_N}


##### Instructions #####

Based on the above context descriptions from {num_contexts} different tasks/domains, derive a set of exactly {num_features} categorical features and {num_features} numerical features that are universally applicable to ALL these tasks. The SAME features will be used for EVERY task, so they must work equally well across all described contexts. Features can be anything from very specific (e.g., occurrences of a specific word) to very abstract (e.g., describing the logical flow of text). Features should capture fundamental properties of text that are meaningful for classification across all the described tasks and generalize even to new and previously unseen examples that may come from a slightly different domain.

IMPORTANT: The SAME set of features will be used for ALL {num_contexts} tasks/contexts simultaneously. Therefore, generate features that are universally applicable and meaningful across all described contexts. Do NOT prioritize any single context -- instead, find features that capture fundamental text properties relevant to classification in all the described domains.

##### Categorical Features #####

Generate exactly {num_features} categorical features. Each feature should be categorical with at least 2 but no more than 5 different possible values. Possible values should be words but not numbers. For any given example, each feature should unambiguously take exactly one value out of the list of possible values. Thus, avoid features that could take two or more values for any example and ensure that there is always exactly one value that fits an example (if necessary, include fallback values such as 'Other', 'Neutral', or 'Not applicable').

##### Numerical Features #####

Generate exactly {num_features} numerical features. Each feature should be numeric and measured on a scale of 0 to 10. Therefore, indicate for each feature what 0 and 10 mean, respectively (e.g., 0 = "few", 10 = "many").

##### Output Format #####
```

```
Return the features as a JSON object
↪  with the following structure:

```json
{
  "categorical_features": [
    {
      "name": "concise name of the
      ↪  feature",
      "possible_values": ["value1",
      ↪  "value2", "value3"],
      "description": "short description
      ↪  of the meaning of this
      ↪  feature"
    }
  ],
  "numerical_features": [
    {
      "name": "concise name of the
      ↪  feature",
      "zero": "meaning of feature value
      ↪  of 0",
      "ten": "meaning of feature value
      ↪  of 10",
      "description": "short description
      ↪  of the meaning of this
      ↪  feature"
    }
  ]
}
```
```

D.1.2 Feature Value Extraction

We use the following single prompt to extract the feature values given an example and task description for all the 5 tasks.

System Prompt:

```
You are a data annotator. Your task is
↪  to annotate a given example by
↪  scoring it with regards to several
↪  criteria. The concrete context is
↪  the following: {context}
```

User Prompt:

```


##### Here is an example that you
↪  should annotate #####

{example}


##### Instructions #####

Score the above example along the
↪  following list of criteria. For
↪  categorical features, assign one of
↪  the possible values. For numerical
↪  features, assign an integer value
↪  from 0 to 10 that most accurately
↪  describes the example.
```

```
The output should be a markdown code
↪  snippet formatted in the following
↪  schema, including the leading and
↪  trailing "\`\`\`json" and "\`\`\`":

```json
{
    "textlengthcategory": string  //
    ↪  'the approximate length of the
    ↪  text' (value can be any of
    ↪  ['Short', 'Medium', 'Long',
    ↪  'VeryLong'])
    "punctuationdensity": string  //
    ↪  'density of punctuation marks
    ↪  in the text' (value can be any
    ↪  of ['Low', 'Medium', 'High'])
    "lexicaldiversity": string  //
    ↪  'variety of unique words in the
    ↪  text, relative to text length'
    ↪  (value can be any of ['Low',
    ↪  'Medium', 'High'])
    "emotionality": string  //
    ↪  'presence of emotionally
    ↪  charged words' (value can be
    ↪  any of ['Neutral',
    ↪  'Emotional'])
    "subjectivitylevel": string  //
    ↪  'the level of subjectivity in
    ↪  the text, inferred from
    ↪  language use' (value can be any
    ↪  of ['Objective', 'Subjective'])
    "useofsuperlatives": string  //
    ↪  'frequency of superlative
    ↪  adjectives and adverbs' (value
    ↪  can be any of ['None', 'Few',
    ↪  'Many'])
    "spellingandgrammarquality": string
    ↪  // 'the number and severity of
    ↪  spelling and grammar mistakes'
    ↪  (value can be any of
    ↪  ['Correct', 'MinorErrors',
    ↪  'MajorErrors'])
    "readability": string  // 'level of
    ↪  readability assessed by
    ↪  language complexity' (value can
    ↪  be any of ['Simple',
    ↪  'Intermediate', 'Complex'])
    "formalitytone": string  // 'the
    ↪  formality of the tone in the
    ↪  text' (value can be any of
    ↪  ['Informal', 'Neutral',
    ↪  'Formal'])
    "useoffirstperson": string  //
    ↪  'presence of the first-person
    ↪  perspective in the text' (value
    ↪  can be any of ['Absent',
    ↪  'Present'])
    "wordcount": int  // 'Total count
    ↪  of words.' (value from 0 to 10,
    ↪  0 means 'No words', 10 means
    ↪  'Very long text (>1000 words)')
    "uniquewordrate": int  // 'Ratio of
    ↪  unique words to total words.'
    ↪  (value from 0 to 10, 0 means
    ↪  'No unique words', 10 means
    ↪  'All words are unique')
```
```

```
    "sentimentscore": int  // 'Overall
    ↪  sentiment of the text, derived
    ↪  from sentiment analysis.'
    ↪  (value from 0 to 10, 0 means
    ↪  'Extremely negative sentiment',
    ↪  10 means 'Extremely positive
    ↪  sentiment')
    "punctuationrate": int  //
    ↪  'Frequency of punctuation marks
    ↪  relative to the number of
    ↪  words.' (value from 0 to 10, 0
    ↪  means 'No punctuation', 10
    ↪  means 'Heavy use of
    ↪  punctuation')
    "stopwordfrequency": int  //
    ↪  'Frequency of common stop words
    ↪  (e.g., the, is, at).' (value
    ↪  from 0 to 10, 0 means 'No stop
    ↪  words', 10 means 'Text composed
    ↪  mostly of stop words')
    "readabilityindex": int  //
    ↪  'Readability of text,
    ↪  quantified by a standard
    ↪  readability index (e.g.,
    ↪  Flesch-Kincaid).' (value from 0
    ↪  to 10, 0 means 'Extremely
    ↪  difficult to read', 10 means
    ↪  'Extremely easy to read')
    "averagewordlength": int  //
    ↪  'Average length of words in the
    ↪  text.' (value from 0 to 10, 0
    ↪  means 'Very short words (<2
    ↪  characters on average)', 10
    ↪  means 'Very long words (>7
    ↪  characters on average)')
    "nounphrasedensity": int  //
    ↪  'Density of noun phrases in the
    ↪  text.' (value from 0 to 10, 0
    ↪  means 'No noun phrases', 10
    ↪  means 'High density of noun
    ↪  phrases')
    "verbosityindex": int  // 'Degree
    ↪  of verbosity or redundancy in
    ↪  language.' (value from 0 to 10,
    ↪  0 means 'Highly terse (no
    ↪  redundancy)', 10 means 'Highly
    ↪  verbose (very redundant)')
    "emotionalityscore": int  //
    ↪  'Presence and intensity of
    ↪  emotional language.' (value
    ↪  from 0 to 10, 0 means
    ↪  'Emotionally neutral', 10 means
    ↪  'Highly emotional')
}
```

Do not add any comments (starting with
↪ //) inside the JSON!
```

### D.1.3 Default Parameters

Following the protocol in (Malberg et al., 2024), we evaluate on 100 training and 100 test examples per dataset (averaged over 3 random draws). For FELIX we run the original multi-stage feature-engineering pipeline, whereas for SAJA we use a fixed rubric prompt (single call) and train the lightweight calibration head on the same 100 training examples. In Table 8, some parameters for LLM calls are shown.

Table 8: Default parameters for FELIX tasks experiments.

| Parameter | Value |
|---|---|
| LLM (gen/scoring) | GPT-4 |
| Temp (gen) | 1.0 |
| Temp (scoring) | 0.0 |
| Features per type | 10 |

### D.1.4 With different task-specific prompts

We now compare the performance of SAJA when the prompts are different for different classification tasks. In Table 9, SAJA (with task-specific prompts) outperforms the published FELIX results and SAJA (with a single prompt for all the tasks) compete effectively with the FELIX results. Overall, the table demonstrates that feature engineering may be an overkill in many classification datasets. Shown below is the task specific user prompt.

**System Prompt:**

```
You are a data scientist. Your goal is
↪  to engineer a set of features
↪  suitable for analyzing and
↪  classifying text data across
↪  different contexts and domains. The
↪  features you generate will be
↪  specifically used for the following
↪  target context: {target_context}
```

User Prompt:

```
##### Instructions #####

Based on the above context description,
↪  derive a set of exactly 5 numerical
↪  and 5 categorical features that
↪  would be useful for analyzing and
↪  classifying text data in this
↪  context. Features can be anything
↪  from very specific (e.g.,
↪  occurrences of a specific word) to
↪  very abstract (e.g., describing the
↪  logical flow of text). However,
↪  features should generalize even to
↪  new and previously unseen examples
↪  that may come from a slightly
↪  different domain.

For categorical features: Each feature
↪  must have 2-5 possible word values.
↪  Each example must unambiguously map
↪  to exactly one value (fallback
↪  values like "Other" or "Not
↪  applicable" are encouraged).
```
```

Table 9: Both SAJA (multi), that leverages task-specific prompts, and SAJA (single prompt), which uses a single prompt compete effectively with the FELIX method on the classification tasks. SAJA (single) numbers are same as those in Table 2. Best results are shown in bold.

Dataset	**Published**	**SAJA** (multi)	**SAJA** (single)
Fake News	**0.96**	**0.96**	**0.96**
Hate Speech	**0.87**	0.82	**0.87**
Paper Reviews	0.93	**0.94**	0.89
Amazon and Yelp	0.96	**0.98**	0.97
Tweet Sentiment	0.84	**0.86**	0.84

```
For numerical features: Each feature is
↪  scored on a 0-10 scale with
↪  explicit semantic anchors (e.g., 0
↪  = "few", 10 = "many").

##### Output Format #####

Return the features as a JSON object
↪  with the following structure:

```json
{
  "categorical_features": [
    {
      "name": "concise name of the
      ↪  feature",
      "possible_values": ["value1",
      ↪  "value2", "value3"],
      "description": "short description
      ↪  of the meaning of this
      ↪  feature"
    }
  ],
  "numerical_features": [
    {
      "name": "concise name of the
      ↪  feature",
      "zero": "meaning of feature value
      ↪  of 0",
      "ten": "meaning of feature value
      ↪  of 10",
      "description": "short description
      ↪  of the meaning of this
      ↪  feature"
    }
  ]
}
```

E Additional Details for Sections 5.2 and 6.3

E.1 MT-Bench Setting

Experimental Setup. Using the shared scoring prompt, we extract seven dimensions (helpfulness, accuracy, coherence, fluency, safety, engagement, overall) for both responses, yielding 14 standardized Likert-scale features (1–5) per pair. We first create a group-aware 80/20 train/test split by `question_id` to prevent leakage across splits.

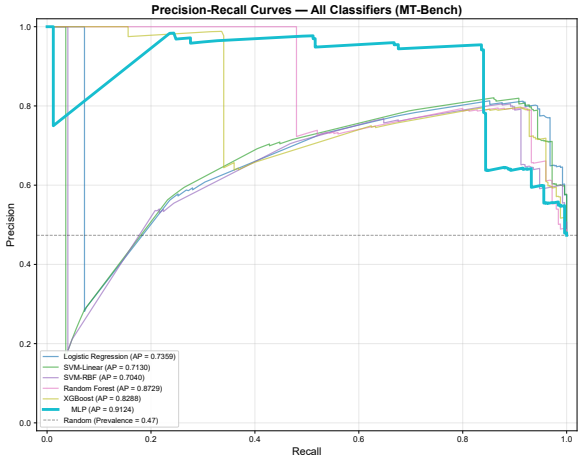

Figure 3: SAJA: precision–recall curves for MT-Bench preference prediction with different classifiers with Fine-grained rubrics. MLP attains average precision (AP) = 0.92 and SVM-Linear attains AP = 0.71.

Within the 80% training portion, we tune hyperparameters using 5-fold `GroupKFold` cross-validation (grouped by `question_id`) and select the best setting based only on training-CV performance. We then retrain the model on the full 80% training set with the selected hyperparameters and report all metrics on the held-out 20% test set.

We repeat this entire procedure independently for the Broad and Fine-Grained rubric variants, using the corresponding feature sets, and report test-set results separately for each variant (Appendix F)

Models and baselines. For MT-Bench pairwise preference, we treat 'Response A is preferred over Response B' as the positive label, 'Response B is preferred over Response A' as the negative label, and ignore the ties. We report accuracy and F1 computed with an explicit positive-label convention and we compute AUC from the predicted preference probabilities prior to thresholding. When a probability threshold is required for accuracy/F1, we use a fixed threshold chosen on the validation set (criterion: maximize validation F1) and report the resulting test performance.

We compare lightweight calibration heads including Logistic Regression (LR), Support Vector Machines (SVMs) with Linear and RBF kernels, Random Forests (RF), XGBoost, and a multilayer perceptron (MLP). Figures 3 and 4 show the performance of the different classifiers. See also Table 10. SVM-Linear consistently gives better scores across different metrics.
```

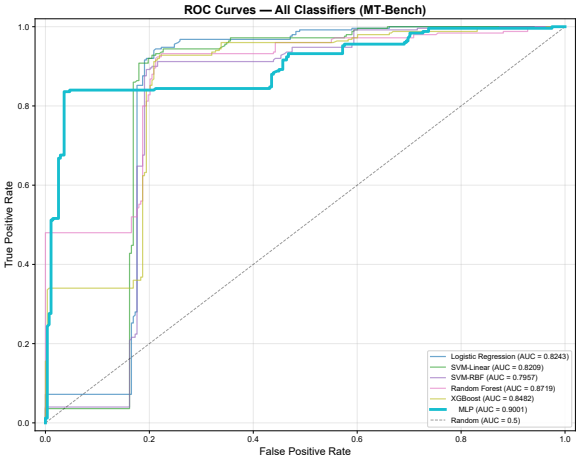

Figure 4: SAJA: AUC-ROC curves for MT-Bench preference prediction with different classifiers with Fine-grained rubrics. MLP achieves AUC of 0.90 with F1 $= 0.58$. SVM-Linear achieves AUC of 0.82 with F1 $= 0.86$.

Budget Analysis Plots. Figure 5 shows AUC and accuracy of the classifiers improve with the number of training examples.

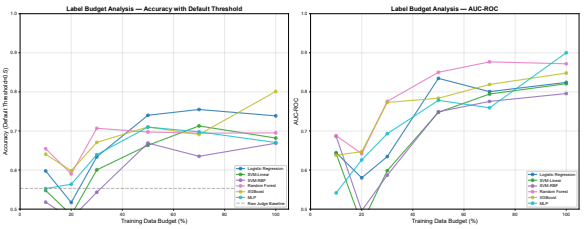

Figure 5: Label Budget plots for MT-Bench preference prediction.

E.2 SummEval Setting

Experimental Setup. SummEval contains news summaries annotated by human raters on multiple quality aspects (e.g., coherence, consistency/factuality, fluency, relevance) as well as an overall quality score. Using the shared scoring rubric prompt, we extract aspect-wise rubric scores for each summary and train a lightweight regression head to predict the overall human rating. We report Pearson r, Spearman ρ, Kendall τ, and RMSE to capture both rank alignment and absolute score calibration.

Models and baselines. We evaluate several regression heads over the rubric features: XGBoost, Ridge regression, SVR, Random Forest, and an MLP. We include two baselines: (i) the Raw LLM Judge (uncalibrated overall score), (ii) a 1-D Isotonic calibration on the overall score.

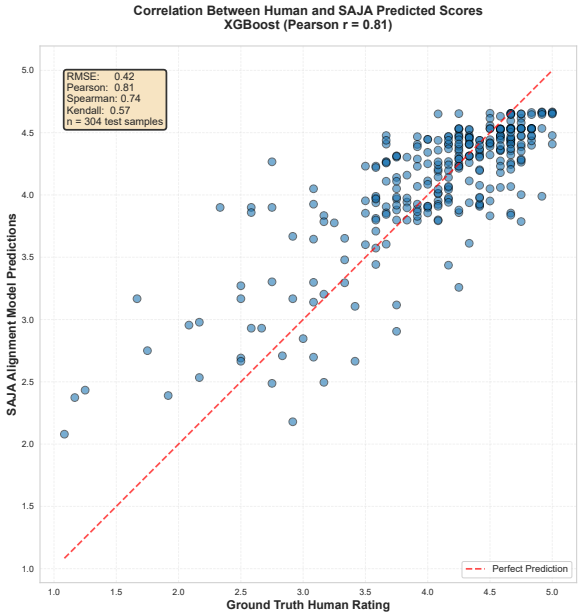

Figure 6: SummEval: Positive correlation between human ratings and SAJA (XGBoost) predicted scores.

Correlation scatter plot. Figure 6 plots predicted overall scores versus human overall ratings for the best-performing SAJA model. The strong diagonal trend indicates that calibrated rubric features track human scores closely; the figure reports a high Pearson correlation (e.g., $r \approx 0.81$) and low error (RMSE ≈ 0.42), while the uncalibrated judge exhibits substantially higher error (RMSE $= 1.46$). This visualization complements rank metrics by showing improved score calibration, not only improved ordering.

Learning curve / label-budget analysis. Figure 7 shows that performance improves with more labeled summaries: RMSE decreases and Pearson correlation increases as training size grows, with the raw judge serving as a high-error reference point. At 20% of training data, we achieve $r = 0.78$ compared to $r = 0.81$ at full budget (100%), indicating that a small labeled subset recovers most of the achievable alignment signal.

F Broad vs Fine-grained Rubrics

Tables 10 and 11 suggest that the value of rubric granularity is task- and objective-dependent. On MT-Bench, even the broad rubric signal is already highly predictive of pairwise preferences and the best-performing results are not uniformly improved by adding finer detail. Fine-grained rubrics show mixed effects across calibration heads: some models improve in specific metrics while others de-

Table 10: Classifier comparison MTBench setting: Fine-grained vs. Broad rubrics.

	Fine-grained			Broad		
Method	**Acc.**	**F1**	**AUC**	**Acc.**	**F1**	**AUC**
Raw Judge (Overall)	0.78	0.78	NA	0.78	0.78	NA
Isotonic (Diff of Overall)	0.60	0.37	0.50	0.60	0.37	0.50
Platt (Diff of Overall)	0.78	0.78	0.88	0.78	0.78	0.88
SAJA (LR)	0.69	0.66	0.82	0.75	0.75	0.91
SAJA (SVM-Linear)	**0.86**	**0.86**	0.82	0.76	0.76	**0.92**
SAJA (SVM-RBF)	0.72	0.71	0.80	0.65	0.65	0.88
SAJA (Rand. Forest)	0.75	0.74	0.87	0.74	0.74	0.75
SAJA (XGBoost)	0.80	0.80	0.85	0.61	0.61	0.82
SAJA (MLP)	0.62	0.58	0.90	0.62	0.61	0.82

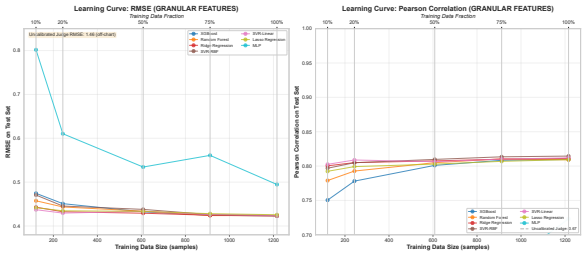

Figure 7: Learning curve plots of different regressors on SummEval.

grade, indicating that additional feature detail can help but also introduces variance/noise and a harder high-dimensional learning problem. In contrast, on SummEval, fine-grained rubrics more consistently strengthen rank alignment: the best regressors achieve higher ρ and τ with fine-grained signals than with broad rubrics, implying that richer aspect-level information helps recover the human ordering more faithfully. Meanwhile, RMSE gaps across granularities are comparatively smaller, suggesting that most absolute error reduction comes from calibration itself, while finer rubrics primarily improve the ranking quality rather than the overall scale fit.

G Feature Importance Analysis

Because we reuse the same rubric feature space across MT-Bench and SummEval, the shift in the top feature importances indicates that SAJA adapts primarily through the calibration head by reweighting the same interpretable dimensions, rather than changing the prompt. On MT-Bench (pairwise preference; Figure 8), feature attribution concentrates on coherence and helpfulness (with engagement also among the most influential dimensions), consistent with conversational preferences being driven by perceived utility and interaction. In contrast, on SummEval (summary quality regression; Figure 9), the most important dimensions are accuracy, coherence, and overall, suggesting that faithfulness and holistic quality dominate human ratings for summaries. Taken together, these patterns support two conclusions: the rubric dimensions are interpretable and diagnostically meaningful, and a single rubric prompt can generalize across tasks while the calibrator learns task-specific priorities by reweighting dimensions.

H Multi-Model Results

Table 6 in the main paper summarizes cross-model results. Below we provide the full breakdown across all benchmarks, models, and rubric variants. Three patterns emerge consistently: (i) SAJA improves over the raw baseline for every model on every benchmark; (ii) weaker models with SAJA can match or surpass stronger models without it; and (iii) rubric granularity interacts with model capability.

SummEval. SAJA reduces RMSE and improves rank correlation for every model (Table 12). The most striking result is that Qwen-9B + SAJA (fine-grained, ρ=0.70) exceeds raw GPT-4.1 (ρ=0.60), demonstrating that calibration can compensate for a weaker judge. GPT-4.1-Nano, the smallest model, improves ρ from 0.50 to 0.59 (+0.09), confirming that even low-capacity judges produce rubric features the calibration head can exploit. Rubric granularity matters more for stronger models. GPT-4.1 gains $\rho = +0.07$ from broad to fine-grained, while GPT-4.1-Mini shows diminishing returns (ρ 0.70 broad vs. 0.71 fine-grained). This suggests that finer rubrics help most when the judge can reliably produce differentiated sub-scores.

MT-Bench. On MT-Bench (Table 13), GPT-4.1 benefits strongly from fine-grained features (Acc/F1: 0.78→0.86), an 8-point gain. Qwen-9B im-

Table 11: Regression comparison on SummEval: Fine-grained vs. Broad rubrics.

	Fine-grained				Broad			
Method	$r\uparrow$	$\rho\uparrow$	$\tau\uparrow$	**RMSE** $\downarrow$	$r\uparrow$	$\rho\uparrow$	$\tau\uparrow$	**RMSE** $\downarrow$
Raw LLM Judge	0.67	0.60	0.50	1.46	0.67	0.60	0.50	1.46
Isotonic (overall)	0.70	0.60	0.50	0.52	0.70	0.60	0.50	0.52
SAJA (XGBoost)	**0.81**	**0.74**	**0.57**	0.42	0.78	0.66	0.50	0.45
SAJA (Ridge)	0.81	0.74	0.56	0.42	0.75	0.67	0.52	0.48
SAJA (SVR)	0.81	0.74	0.56	0.42	0.78	0.66	0.51	0.46
SAJA (Rand. Forest)	0.81	0.74	0.57	0.42	0.78	0.65	0.50	0.45
SAJA (MLP)	0.74	0.67	0.50	**0.36**	0.76	0.62	0.48	0.48

Table 12: SummEval results across models and rubric variants.

Model	**Rubric**	**RMSE** $\downarrow$	$r\uparrow$	$\rho\uparrow$	τ
GPT-4.1	Baseline	1.46	0.67	0.60	0.50
GPT-4.1	Broad	0.45	0.78	0.65	0.50
GPT-4.1	Fine-grained	**0.42**	**0.81**	**0.72**	**0.55**
GPT-4.1-Mini	Baseline	1.18	0.71	0.65	0.54
GPT-4.1-Mini	Broad	**0.46**	**0.80**	**0.70**	0.54
GPT-4.1-Mini	Fine-grained	0.48	0.79	0.71	0.53
GPT-4.1-Nano	Baseline	1.32	0.54	0.50	0.41
GPT-4.1-Nano	Broad	0.53	0.67	0.57	0.43
GPT-4.1-Nano	Fine-grained	**0.51**	**0.71**	**0.59**	**0.43**
Qwen-9B	Baseline	1.37	0.60	0.62	0.48
Qwen-9B	Broad	0.50	**0.76**	0.67	0.50
Qwen-9B	Fine-grained	0.52	0.74	**0.70**	**0.51**

Table 13: MT-Bench results across models and rubric variants.

Model	**Rubric**	**Acc.**	**F1**	**AUC**
GPT-4.1	Baseline	0.78	0.78	–
GPT-4.1	Broad	0.76	0.76	0.92
GPT-4.1	Fine-grained	**0.86**	**0.86**	0.82
Qwen-9B	Baseline	0.64	0.63	–
Qwen-9B	Broad	**0.67**	**0.67**	0.89
Qwen-9B	Fine-grained	0.70	0.66	0.67

proves from 0.64/0.63 to 0.67/0.67 with the broad rubric. Interestingly, for Qwen-9B the fine-grained rubric improves accuracy (0.70) but slightly reduces F1 (0.66), with AUC dropping from 0.89 to 0.67. This suggests that smaller models may not produce reliable sub-criterion scores, introducing noise that hurts the calibration head's probability estimates while still improving point predictions.

Table 14: LLM-RUBRIC binary classification across models.

Model	**Base. Acc.**	**Base. F1**	**SAJA F1**	**SAJA AUC**
GPT-4.1	0.64	0.57	**0.62**	0.61
Qwen-9B	0.61	0.58	**0.62**	0.58

LLM-RUBRIC. On LLM-RUBRIC (Table 14), both GPT-4.1 and Qwen-9B reach identical calibrated F1 of 0.62, despite different baseline performance (0.57 vs. 0.58). This convergence suggests that when the rubric captures the relevant quality dimensions, the calibration head can extract the same alignment signal regardless of judge scale. GPT-4.1 retains a slight AUC advantage (0.61 vs. 0.58), indicating marginally better-calibrated probability estimates.

I Prompt Consolidation Ablation

To validate SAJA's core assumption that consolidating all rubric questions into a single prompt does not degrade score quality, we compare per-question prompting (one rubric question per LLM call, as in LLM-RUBRIC) against a unified single prompt (all questions in one call, as in SAJA). Both settings use argmax-only judge outputs on the LLM-RUBRIC dataset (Q0: overall quality), with rubric content, decoding, splits, and calibration protocol held fixed.

Table 15: Prompt consolidation ablation on LLM-RUBRIC (Q0). Only the prompting format varies; both use argmax-only outputs.

Method	r	ρ	τ
Per-question Prompt	0.29	0.28	0.22
Unified Prompt (GPT-4.1)	0.25	0.28	0.21
Unified Prompt (GPT-4.1-mini)	0.24	0.27	0.21

Spearman ρ is identical (0.28 vs. 0.28) and Kendall τ nearly identical (0.21 vs. 0.22), with a small drop in Pearson r (0.25 vs. 0.29). Results hold across models, confirming the finding is not model-specific. The small Pearson r gap reflects sensitivity to absolute score magnitude, while rank-order metrics (ρ, τ), which matter most for calibration, are preserved. This validates SAJA's core design: asking all rubric questions at once incurs no

meaningful quality loss while yielding $K\times$ savings in API calls and tokens (quantified in Section 6.4).

J Human Agreement Ceiling

To contextualize SAJA's correlation numbers, we compute human-human agreement ceilings on benchmarks that provide per-rater annotations.

SummEval. Using the 3 expert annotations per summary, we compute Spearman ρ between each expert's ratings and the mean of the remaining experts (leave-one-out). The mean ceiling is $\rho = 0.758$. SAJA achieves $\rho = 0.74$, which is **97.6%** of the human ceiling. The average pairwise expert-to-expert agreement is $\rho = 0.634$, meaning SAJA exceeds the level at which individual experts agree with each other.

Table 16: SummEval human agreement ceiling (3 experts) and SAJA.

Metric	ρ
Pairwise expert–expert	0.634
Expert-to-consensus (leave-one-out)	0.758
SAJA	0.740

LLM-RUBRIC. Inter-annotator agreement on LLM-RUBRIC is substantially lower: pairwise $\rho = 0.070$, individual-to-consensus $\rho = 0.088$. This reflects the inherent difficulty of ordinal quality scoring on dialogue responses. SAJA ($\rho = 0.27$) exceeds this ceiling because SAJA models individual annotator preferences, whereas the ceiling measures blind prediction of the average rating.

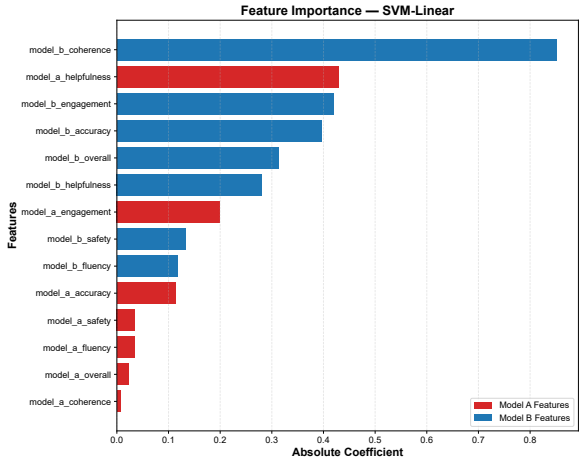

Figure 8: Feature importance on MT-Bench. The top dimensions are coherence and helpfulness.

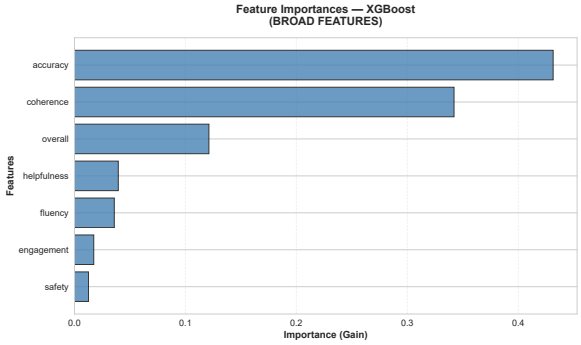

Figure 9: Feature importance on SummEval. Top features are accuracy and coherence.

K Statistical Significance and Variance

To address concerns about whether performance differences are within sampling noise, we report results across 10 seeds (41–50) with standard deviations on the FELIX classification benchmarks.

Table 17: SAJA vs FELIX classification F1. 3-seed columns follow FELIX's evaluation protocol. 10-seed results confirm stability.

Dataset	**FELIX**	**SAJA**	**SAJA-10s**	**SD-10s**
Fake news	0.96	0.96	0.95	0.02
Hate speech	0.87	0.87	0.84	0.05
Papers	0.93	0.89	0.88	0.03
Reviews	0.96	0.97	0.97	0.02
Sentiment	0.84	0.84	0.78	0.06
Average	**0.91**	**0.91**	**0.88**	**0.036**

One-sample t-tests comparing SAJA's per-seed F1 (seeds 41–43) against FELIX's reported mean confirm that none of the differences are statistically significant at $\alpha = 0.05$ (all $p > 0.05$). Even for Papers, where SAJA scores 4 points below FELIX, $p = 0.08$.

Table 17 reveals that Fake news (0.95 ± 0.02), Reviews (0.97 ± 0.02), and Papers (0.88 ± 0.03) are highly stable across seeds. Sentiment (0.78 ± 0.06) and Hate speech (0.84 ± 0.05) show higher variance, reflecting that these tasks have noisier label boundaries at the 100-example scale. The 3-seed average (0.91) is slightly higher than the 10-seed average (0.88), which is expected: the original 3 seeds (41–43) were selected to match FELIX's protocol, and additional seeds introduce more challenging splits. Importantly, FELIX's published 0.91 is itself a 3-seed point estimate without reported variance, so both methods operate under the same uncertainty regime.

Table 18 shows that all p-values comfortably ex-

Table 18: One-sample t-tests: SAJA (3 seeds) vs. FELIX mean F1.

Dataset	**Diff**	t	p
Fake news	+0.00	0.23	0.84
Hate speech	−0.00	−0.10	0.93
Papers	−0.04	−3.25	0.08
Reviews	+0.01	0.61	0.60
Sentiment	+0.00	0.00	1.00

ceed 0.05. The closest case is Papers (p=0.08), where SAJA's generic rubric does not capture the specialized features that FELIX's iterative discovery finds for this domain. For the remaining four tasks, differences are negligible ($|\text{Diff}| \leq 0.01$), confirming that SAJA's single-call, fixed-rubric design achieves parity with FELIX's multi-call, per-task feature engineering.