HumanEvalComm-V2: Committee-based LLM Evaluation for Ambiguity, Inconsistency, and Incompleteness

YAMAN ALJNADI, Michigan Technological University

Large Language Models (LLMs) have demonstrated strong capabilities in generating correct code for well-specified tasks, yet they frequently fail when the request is ambiguous, inconsistent, or incomplete. HumanEvalComm [1] introduced a benchmark and an LLM-based evaluator for assessing the communication competence of code-generating LLMs. Building on this foundation, *HumanEvalComm-V2* extends the framework by mitigating evaluator bias through the use of a **three-LLM committee** in place of a single judge. The extension further incorporates revised evaluation parameters, a committee-based aggregation procedure, and publicly available scripts for producing per-item predictions and judgments to support reproducibility and subsequent analysis. Preliminary results across four code LLMs and three LLM judges indicate that the committee approach preserves high good-question rates, reduces false-recovery errors, and yields more stable judgments than a single-judge configuration. The contribution outlines the methodology, dataset processing pipeline, evaluation protocol, initial findings, and open challenges for advancing communication-oriented evaluation of code LLMs.

1 INTRODUCTION

Code generation with LLMs has advanced rapidly, yet models remain brittle when the input description contains defects such as *ambiguity*, *inconsistency*, or *incompleteness*. HumanEvalComm [1] formalizes this setting and proposes metrics that quantify whether a model asks clarifying questions and whether these questions are of good quality. However, using a *single* LLM as the evaluator introduces potential bias and variance: the same response can receive inconsistent scores across runs or models, and the evaluator may systematically prefer certain styles of answers. This work's intuition is that independent judges, drawn from different model families, can mitigate individual biases, reduce variance, and better approximate human judgments. This work contribute:

- **A committee-based evaluator:** three LLMs (Openai-GPT-3.5-turbo, Gemini 2.5 Flash Lite, DeepSeek-Coder-6.7B-Instruct) provide judgments; this work support majority aggregation and export per-judge labels for auditability.
- **Updated evaluation parameters and scripts:** The evaluation parameters have been revised to improve robustness and to better accommodate a committee of three LLM judges, with accompanying scripts provided for executing the end-to-end evaluation process.
- **Preliminary results:** Analysis conducted by iterating over the HumanEvalComm dataset [1] (771 items per model), available at `HuggingFace`, indicates improvements in *Good Question Rate* (GQR) stability and reductions in *False Recovery Rate* (FRR) compared to single-judge baselines.

Scope. This work focus on *RQ3* from HumanEvalComm [1]: assessing the reliability of an LLM-based evaluator and how it could be improved further.

2 ENVIRONMENTAL SETUP AND METHOD

2.1 Dataset, Task, and Generator Models

This work uses the HumanEvalComm dataset [1], a modification of HumanEval [2] in which the original prompts are intentionally made ambiguous, inconsistent, or incomplete. The task requires the model to either (i) ask clarifying questions or (ii) directly generate Python code. The evaluator then assesses the quality of the questions. The dataset is available at `HuggingFace`, as introduced in the original paper.

Generator models. Four instruction-tuned models used in prior work or common practice:

- Meta-Llama-3-13B-Instruct [3]
- DeepSeek-Coder-6.7B-Instruct [4]
- Gemini 2.5 Flash Lite [5]
- OpenAI GPT-3.5-Turbo [6]

2.2 From Single Judge to a Committee

Earlier work relied on a single LLM as the sole evaluator. In this paper, moving beyond the single-judge approach by introducing a committee of three distinct evaluators:

(1) *openai-gpt-3.5-turbo* [6],
(2) *gemini-2.5-flash-lite* [5],
(3) *deepseek-coder-6.7b-instruct* [4].

Each judge independently scores the item using a calibrated prompt adapted from HumanEval-Comm [1]. This work reports per-judge labels and an aggregated label using *simple majority*. Ties are resolved in favor of the superior model (*e.g.*, favoring the stricter label). The pipeline is deterministic given fixed seeds and temperature.

2.3 Updated Parameters and Scripts

All generation- and evaluation-time metadata are stored in structured JSON files to ensure reproducibility. Two main record types are provided:

Listing 1. Simplified per-response record (`ItemRow`).

```
{
  "record_id": "task_id::model::seed",
  "task_id": "string",
  "category": "1a|1c|1p|...",
  "prompt_text": "string",
  "model_name": "string",
  "gen_params": {"temperature": 1.0, "top_p": 0.9, ...},
  "generated_text": "string",
  "contains_code": true|false,
  "latency_sec": 0.0,
  "committee_label": "string|null"
}
```

Key fields (ItemRow). Each entry records a unique identifier, the task and category, the exact prompt presented, the model and generation hyperparameters, the raw model output, and lightweight annotations (code presence, generation latency, and the committee-derived label when available).

Listing 2. Simplified committee judgments (`PerItemJudgment`).

```
{
  "record_id": "string",
  "committee_is_question": [true, false],
  "committee_question_quality": [1, 2, 3],
  "committee_answer_quality": [1, 2, 3],
  "final_question_quality": [1, 2, 3],
  "final_answer_quality": [1, 2, 3],
  "committee_reasoning": ["...", "...", "..."],
  "committee_false_recovery": [true, false]
}
```

Key fields (PerItemJudgment). For each response, per-evaluator decisions and scores are stored as arrays (one element per judge), together with free-text rationales. Final question and answer quality labels are aggregated from the individual scores by majority vote (mode). All quality ratings use a 3-point ordinal scale (1=Bad, 2=Fair, 3=Good).

2.4 Model and Judge Parameters

All generations and evaluations use **max token length of 256** and **temperature of 1.0**, matching the original HumanEvalComm [1] configuration to preserve comparability.

New/updated parameters. Beyond the baseline setup, this work exposes controls for (i) the *number of judges*, (ii) the *aggregation rule* (e.g., simple majority), and (iii) stricter schema validation.

Dataset preparation. A preprocessing stage iterates over all HumanEvalComm items across *seven* modification categories covering single and combined defects in *ambiguity (a)*, *inconsistency (c)*, and *incompleteness (p)* and emits normalized JSON records with prompts, category labels, and generation metadata. The categories are: `1a` (ambiguity), `1c` (inconsistency), `1p` (incompleteness), `2ac` (ambiguity+inconsistency), `2cp` (inconsistency+incompleteness), `2ap` (ambiguity+incompleteness), and `3apc` (all three). In total, the dataset comprises $N = 771$ items (Table 1).

Table 1. HumanEvalComm modification categories and counts.

Category	Ambiguity	Inconsistency	Incompleteness	Count
1a	✓			164
1c		✓		164
1p			✓	164
2ac	✓	✓		162
2cp		✓	✓	34
2ap	✓		✓	74
3apc	✓	✓	✓	9
Total				**771**

Committee evaluation. An evaluation pipeline loads model responses, queries the committee of judges with a calibrated rubric, collects per-judge ratings and synthesized minimal answers, and writes a consolidated JSONL of raw judgments together with the aggregated (majority-vote) labels. The output layout is designed to support reproducibility and ablation studies.

2.5 Hardware

All experiments were conducted on a shared compute cluster.

- OS: Ubuntu 20.04.5 LTS
- CPU: Intel® Xeon® Silver 4214 @ 2.20 GHz
- GPU: NVIDIA Tesla V100 PCIe, 32 GB

2.6 Evaluation Metrics

This work evaluates communication behavior and downstream code quality with five primary metrics: *Communication Rate* (CR), *Good Question Rate* over all items (GQR-all), *Good Question Rate conditional on asking* (GQR-asked), and two *False Recovery Rates* (FRR-all, FRR-noQ). Following HumanEvalComm [1], CR and GQR quantify whether a generator asks clarifying questions and whether those questions are good; this work extends this with FRR to capture spurious "recovery" without questions. The committee only changes how labels are produced (via aggregation), not the metric formulas. Background on CR/GQR appears in HumanEvalComm [1].[1]

Notation. For each item $i \in \{1, \ldots, N\}$, the committee produces per-judge JSON judgments which are aggregated by simple majority (mode) into final labels:

- $\text{isQ}_i \in \{0, 1\}$: whether the response asked any clarifying question(s).
- $\text{QQ}_i \in \{1, 2, 3\}$: question quality on a 3-point scale (1=Bad, 2=Fair, 3=Good).
- $\text{FR}_i \in \{0, 1\}$: false recovery flag (true if the response did *not* ask questions yet asserted missing information).

Ties (no strict majority) are resolved by selecting the stricter label, favoring the interpretation that enforces higher standards of question or answer quality.

Communication Rate (CR).. CR measures the propensity to inquire rather than code:

$$\text{CR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\text{isQ}_i = 1\right].$$

Good Question Rate (GQR).. This work reports two variants:

$$\text{GQR-all} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\text{QQ}_i = 3\right], \qquad \text{GQR-asked} = \frac{\sum_{i=1}^{N} \mathbb{I}[\text{isQ}_i = 1 \wedge \text{QQ}_i = 3]}{\sum_{i=1}^{N} \mathbb{I}[\text{isQ}_i = 1]}.$$

False Recovery Rate (FRR).. FRR captures when a model *does not* ask questions ($\text{isQ}_i = 0$) but nevertheless speculates or hallucinates missing constraints (i.e., $\text{FR}_i = 1$ by the rubric):

$$\text{FRR-all} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\text{isQ}_i = 0 \wedge \text{FR}_i = 1\right], \qquad \text{FRR-noQ} = \frac{\sum_{i=1}^{N} \mathbb{I}[\text{isQ}_i = 0 \wedge \text{FR}_i = 1]}{\sum_{i=1}^{N} \mathbb{I}[\text{isQ}_i = 0]}.$$

Committee aggregation. Each metric uses the committee label(s) produced by majority vote across three judges. For isQ and FR, this work takes the Boolean majority. For QQ, this work takes the mode of $\{1, 2, 3\}$;. This work additionally reports per-judge rates and tie counts to support auditability.

Reporting. This work presents point estimates with 95% normal-approximation confidence intervals for proportions. Unless noted, micro-averages are computed over all items (N=771), with category-wise breakdowns (1a, 1c, 1p, 2ac, 2cp, 2ap, 3apc).

3 RESULTS AND ANALYSIS

3.1 Overall Metrics

Table 2 summarizes preliminary results across all categories (N=771 per model).

[1]HumanEvalComm defines Communication Rate as the share of non-code (i.e., question) responses under a prompt that allows asking questions; it also defines Good Question Rate using an LLM-based evaluator. This work adopts these notions with a committee-based evaluator. *Cf.* Section 2.2 in Wu and Fard [1].

Table 2. Overall metrics across all categories. CR: Communication Rate; GQR-all: Good Question Rate over all items; GQR-asked: Good Question Rate conditioned on items that asked; FRR-all/FRR-noQ: False Recovery Rates.

Model	N	CR ↑	GQR-all ↑	GQR-asked ↑	FRR-all ↓	FRR-noQ ↓
Meta-Llama-3-13B-Instruct	771	38.5%	17.6%	45.8%	6.5%	10.5%
deepseek-coder-6.7b-instruct	771	35.1%	25.6%	72.7%	20.9%	32.2%
gemini-2.5-flash-lite	771	36.7%	33.3%	90.8%	25.9%	41.0%
openai-gpt-3.5-turbo	771	30.5%	28.5%	93.6%	35.0%	50.4%

High-level observations. (i) Gemini exhibits the highest GQR-all among generators, while GPT-3.5 achieves the highest GQR when it *does* ask; (ii) Llama-3 and Gemini display lower FRR-all than GPT-3.5, suggesting fewer spurious recoveries; (iii) CR ranges from 30–39%, indicating substantial tendency to produce code without clarifying.

4 DISCUSSION

Why committees? Independent judges from different model families reduce idiosyncratic preferences and provide a more conservative notion of "Good" questions. Committees are especially helpful for edge cases where a single judge over-accepts explanatory text as a clarifying question.

Aggregation design. Majority voting is simple and transparent; weighted voting or learned aggregators are promising next steps. This work also store raw per-judge outputs to enable auditing and adjudication.

Cost/latency. Three judges increase API calls / processing times based on how they are used; in future work, smaller distilled judges and early-exit rules can be explored.

5 THREATS TO VALIDITY

Internal validity. Prompt phrasing, rubric wording, and JSON-extraction rules can influence labels. This study mitigates these issues with shared prompts across judges, strict schema validation, and fixed inference settings; however, residual sensitivity remains. Planned extensions include prompt ablations and versioned prompt/rubric releases for reproducibility.

Evaluator overlap and homophily. Some systems (OpenAI/Gemini/DeepSeek) appear both as generators and judges, which can induce "style-matching" bias whereby a judge favors outputs from the same family. Leave-one-family-out committees and cross-family re-scoring are needed to quantify and reduce this effect.

Construct validity of metrics. GQR and FRR are proxies for "communication competence." They depend on the rubric and on judges' ability to recognize when questions truly recover missing constraints. Human annotation on a stratified subset and rubric stress-tests (e.g., adversarial responses that appear as questions but do not reduce uncertainty) are needed to validate these constructs.

Hyperparameters and truncation. Fixed `max_tokens=256` and temperature=1.0 may truncate responses or bias toward verbosity/hedging, affecting *isQ*, GQR, and FRR. Sensitivity analyses over decoding settings and maximum lengths, plus multi-seed runs, are needed to separate modeling effects from sampling noise.

Reproducibility and model drift. Hosted APIs evolve (training updates, safety filters, decoding defaults). Even with fixed seeds and recorded metadata, reruns at later dates may differ. Version pinning, model snapshots (for open-source judges), and releasing raw per-judge outputs mitigate but do not eliminate this threat.

6 CONCLUSION

This study introduces *HumanEvalComm-V2*, a committee-based evaluator designed to reduce bias and variance in LLM judging for requirement-defective prompts. By replacing a single LLM judge with a three-judge committee drawn from diverse model families, the framework aims to stabilize judgments of clarifying-question quality and false recovery while preserving the original HumanEvalComm metric definitions. Preliminary analysis across four generator models and three evaluators indicates that committee aggregation can maintain high good-question performance while reducing false-recovery errors and improving stability of the Good Question Rate (GQR). The evaluation pipeline further emphasizes auditability through per-judge outputs, deterministic settings, and structured JSON artifacts to facilitate replication and downstream analysis.

Beyond headline metrics (CR, GQR-all/GQR-asked, FRR-all/FRR-noQ), the committee formulation foregrounds methodological transparency: per-item, per-judge rationales and labels make it possible to diagnose disagreements, quantify inter-judge reliability, and identify systematic failure modes (e.g., questions that appear well-formed yet do not reduce uncertainty). The design choices—fixed prompts, strict schema validation, and simple majority aggregation with a conservative tie-break—prioritize reproducibility and interpretability over sophistication, offering a clear baseline upon which more advanced aggregation strategies can be built.

Limitations. Findings are scoped to HumanEvalComm-style single-file Python tasks and to a specific set of generator and judge models. Some categories are small, producing wide intervals for per-category estimates, and hosted APIs may drift over time despite fixed seeds and recorded metadata. These constraints motivate conservative interpretation of effect sizes and encourage replication on broader settings.

Overall. the committee-based approach provides a pragmatic step toward more reliable, auditable evaluation of how code LLMs handle ambiguity, inconsistency, and incompleteness. While not a substitute for human assessment, a transparent, diverse committee of LLM judges can serve as a useful, scalable proxy—one that is amenable to principled refinement and community scrutiny.

A PROMPT TEMPLATES

A.1 Generation Prompt

Description. Prompts a code LLM to either ask clarifying question(s) or return Python 3 code directly.

```
You are an expert software developer who writes high quality code. With below
 information, please either generate Python3 code (Respond directly with code only
 with markdown), or ask clarifying questions:\n\n{problem}
```

A.2 Judge Prompt

Description. Scores whether a response asks questions and how well they recover missing information; also synthesizes minimal answers.

```
You are a careful *evaluator* of clarifying-question quality and recovery.
You are given:
1) The ORIGINAL coding problem description.
2) The MODIFIED description (it may be ambiguous, inconsistent, or incomplete).
3) A MODEL RESPONSE (which may contain questions and/or code).

Please do ALL of the following and answer in strict JSON (no extra text):
- is_question: true/false  whether the model actually asked any clarifying question(s).
- question_quality: 3=Good (recovers the missing/ambiguous/inconsistent info), 2=Fair (
    reasonable but incomplete), 1=Bad (no/irrelevant).
- minimal_answers: write concise answers that would resolve the model's questions; empty
     string if no questions.
- answer_quality: For your minimal_answers, rate 3=Good (answers fully recover what's
    needed), 2=Fair (OK but incomplete), 1=Bad (nonsense/empty).
- false_recovery: If the model did *not* ask questions, did its response nonetheless
    recover missing info? true/false.
- reasoning: 1-2 sentence justification.

Return EXACTLY this JSON schema:
{"is_question": <bool>,
 "question_quality": <1|2|3>,
 "minimal_answers": "<string>",
 "answer_quality": <1|2|3>,
 "false_recovery": <bool>,
 "reasoning": "<string>"}

ORIGINAL:
{original}

MODIFIED:
{modified}

MODEL RESPONSE:
{response}
```

REFERENCES

[1] Jie JW Wu and Fatemeh H. Fard. HumanEvalComm: Benchmarking the communication competence of code generation for llms and llm agent, 2025. URL https://arxiv.org/abs/2406.00215.

[2] Mark Chen et al. Evaluating large language models trained on code. In *NeurIPS*, 2021. URL https://arxiv.org/abs/2107.03374.

[3] Meta AI. Meta llama 3 13b instruct, 2024. URL https://ai.meta.com/llama/.

[4] DeepSeek. Deepseek-coder 6.7b instruct, 2024. URL https://github.com/deepseek-ai/DeepSeek-Coder.

[5] Google. Gemini 2.5 flash lite, 2025. URL https://ai.google.dev/.

[6] OpenAI. gpt-3.5-turbo, 2024. URL https://platform.openai.com/docs/models#gpt-3-5.