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 requirements are ambiguous, inconsistent, or incomplete. HumanEvalComm [6] 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 [6] 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 [6] (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 Wu and Fard [6]: 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 [6], 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. We evaluate four instruction-tuned models used in prior work or common practice:

- Meta-Llama-3-13B-Instruct [1]
- DeepSeek-Coder-6.7B-Instruct [3]
- Gemini 2.5 Flash Lite [4]
- OpenAI GPT-3.5-Turbo [5]

2.2 From Single Judge to a Committee

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

(1) *openai-gpt-3.5-turbo* [5],

(2) *gemini-2.5-flash-lite* [4],

(3) *deepseek-coder-6.7b-instruct* [3].

Each judge independently scores the item using a calibrated prompt adapted from Wu and Fard [6]. This work reports per-judge labels and an aggregated label using *simple majority*. Ties are surfaced explicitly and retained for analysis; in downstream experiments this work either keeps ties as "undecided" or applies a deterministic tie-breaker (*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. We provide two main record types:

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 configuration to preserve comparability.

New/updated parameters. Beyond the baseline setup, we expose 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

- 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

We evaluate 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 HumanEval-Comm [6], CR and GQR quantify whether a generator asks clarifying questions and whether those questions are good; we extend this with FRR to capture spurious "recovery" without questions. Our committee only changes how labels are produced (via aggregation), not the metric formulas. Background on CR/GQR appears in Wu and Fard [6].[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:

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

Ties (no strict majority) are surfaced as *undecided* and excluded from the numerator and denominator of any metric that depends on the tied label; when reported, we disclose tie counts.

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

$$\mathrm{CR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\big[\mathrm{isQ}_i = 1\big] .$$

Good Question Rate (GQR).. We report two variants:

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

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

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

Committee aggregation. Each metric uses the committee label(s) produced by majority vote across three judges. For isQ and FR, we take the Boolean majority. For QQ, we take the mode of $\{1, 2, 3\}$; if all three disagree (e.g., 1, 2, 3), we mark *undecided.* We additionally report per-judge rates and tie counts to support auditability.

Reporting. We present 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) reported in Section **??**. For completeness, we also track *Pass@1* and *Test Pass Rate* to relate communication to end-task correctness, following standard practice.

3 RESULTS AND ANALYSIS

3.1 Overall Metrics

Table 2 summarizes preliminary results across all categories (N=771 per model). Placeholders for figures referencing the full distribution by category are provided below.

[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. We adopt these notions with a committee-based evaluator. *Cf.* Section 2.2 in Wu and Fard [6].

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 we can explore smaller distilled judges and early-exit rules.

5 THREATS TO VALIDITY

Internal validity. Prompt phrasing and extraction rules can influence labels; this work mitigate with strict schema validation and shared prompts across judges.
External validity. Results may not generalize beyond Python/HumanEvalComm; other domains (multi-file repos, non-Python) are future work.
Construct validity. GQR and FRR operationalize "communication competence" but remain proxies. Human studies are needed to calibrate committee behavior against expert judgments.
Model drift. Hosted APIs may change over time; this work record exact versions and temperatures.

6 CONCLUSION

This work introduce HumanEvalComm-V2, a committee-based evaluator that aims to reduce bias and variance in LLM judging for requirement-defective prompts. Early results suggest committees improve robustness without changing metric definitions. This work release scripts and data schemas to facilitate replication, and plan comprehensive ablations on aggregation rules, judge diversity, and per-category behavior.

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] Meta AI. 2024. Meta Llama 3 13B Instruct. https://ai.meta.com/llama/

[2] Mark Chen et al. 2021. Evaluating Large Language Models Trained on Code. In *NeurIPS*. https://arxiv.org/abs/2107.03374

[3] DeepSeek. 2024. DeepSeek-Coder 6.7B Instruct. https://github.com/deepseek-ai/DeepSeek-Coder

[4] Google. 2025. Gemini 2.5 Flash Lite. https://ai.google.dev/

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

[6] Jie JW Wu and Fatemeh H. Fard. 2025. HumanEvalComm: Benchmarking the Communication Competence of Code Generation for LLMs and LLM Agent. arXiv:2406.00215 [cs.SE] https://arxiv.org/abs/2406.00215