First Make It Playable, Then Make It Good: Staged Interaction Learning for Small Dialogue-Game Agents

Anonymous ACL submission

Abstract

We present an anonymous 2B-parameter submission to the LM Playschool Challenge. We use the base model and train small dialogue-game agents using three steps: a) SFT on only successful game trajectories from Playpen, b) weighted turn-level SFT, and c) teacher-guided SFT. The teacher model (which is a larger model) is only used to fix formatting and evaluate examples, but does not create new gold actions. Our final model scores 57.12 clemscore and 42.68 statscore on the public Playpen validation. Our findings suggest that imitating full trajectories helps with playability, while turn-level and teacher-guided training usually improve decision-making and increase the overall score. Alternative procedurally heavy approaches like replay-repair and hard-example mining did not help, which suggests that small models are performant simply by using careful curation strategies rather than aggressive changes.

1 Introduction

Large language models (LLMs) have gained significant popularity as agents in multi-step interactive environments, despite traditionally being trained on isolated instruction-response pairs. Dialogue games, however, demand a distinct form of competence. An effective agent must first observe a state, then produce a valid action, receive subsequent feedback, and interact according to a defined protocol (Chalamalasetti et al., 2023; Beyer et al., 2024; Schlangen et al., 2025). In this context, language serves as both a representational medium and the very mechanism through which the agent acts. As a result a generated string may constitute a legal move, an illegal move, a premature termination, or a failure to adhere to the rules of the game.

Within such setups, a model may fail in various ways: by providing unnecessary explanations, employing incorrect action formats, failing to halt when required, executing multiple actions simultaneously, or breaking the game's established role structure. To address these challenges within the Playpen framework, our approach focuses on equipping the language model with a robust understanding of interaction dynamics and legal rules, rather than merely encouraging the memorisation of specific game states. In the context of this challenge, playability measures the extent to which the model has internalised interactional rules well enough to complete episodes without aborting. Quality, conversely, evaluates how well the model performs the task once valid participation is established.

We aim to develop a model that explicitly captures this distinction. In our dataset, a successful Playpen transcript comprises both interactional signals, which demonstrate adherence to participation rules, and decision signals, which guide the appropriate action to take given a specific dialogue or game state. To leverage these effectively, we employ a staged and incremental training strategy. First, we fine-tune the model on complete, successful trajectories, truncated to a maximum of 1,024 tokens, enabling it to learn the global rhythm of valid participation. Second, we factorise these successful trajectories into local state-action examples (mapping dialogue/game history to next assistant action), thereby aligning the training objective more closely with the inference-time decision problem. Third, we utilise a more capable model as a constrained teacher and this teacher evaluates selected examples and generates invalid 'near-misses' for repair, without inventing new gold-standard actions.

Our architectural design is grounded in established literature on successful and guided learning. Success-filtered training leverages high-quality trajectories as supervision (Zelikman et al., 2022; Gulcehre et al., 2023, inter salia), whilst value-weighted policy learning applies stronger updates to higher-value actions (Peng et al., 2019). Furthermore, cognitive accounts of learning emphasise

guided participation, scaffolding, and the gradual internalisation of rule-governed practices (Vygotsky, 1978; Bruner, 1983). Recent developments in text-based game agents also demonstrate that complex action spaces can be rendered more tractable by representing states and actions as text, decomposing extensive action choices into smaller, manageable decisions (Xu et al., 2025). We synthesise these principles to formulate a robust training recipe for Playpen dialogue games.

Our primary contributions in our submission are: (a) We propose a three-stage interaction-learning recipe for a 2B-parameter Playpen agent; (b) We provide an analysis demonstrating that successful trajectories and factorised state-action examples serve complementary roles in improving both playability and decision quality; (c) We introduce a teacher-guided augmentation stage that evaluates and repairs examples without generating new gold actions; and (d) We finally present compact ablation studies revealing that our lightweight, teacher-guided training recipe performs highly effectively, precluding the need for computationally heavier replay-repair mechanisms, outcome weighting, or hard-example mining.

2 Method

2.1 Stage 1: Success SFT for valid participation

We fine-tune Qwen3.5-2B using only successful Playpen trajectories. We export the Playpen interactions train split to JSONL and filter for episodes where `Outcome` equals `"success"`. We focus on successful episodes because this stage aims to teach the model what a valid interaction looks like. The model, before doing anything, requires understanding and learning the structure of a successful dialogue-game episode before improving its decision-making. This is aligned with the Playpen framing of Dialogue Games (Horst et al., 2025). It is also related to success-filtered post-training methods, which train on model outputs or traces that satisfy an external success criterion (Zelikman et al., 2022).

We format each successful episode with the Qwen chat template and train the model using standard causal language model loss, stopping at 1,024 tokens. We chose this limit to balance coverage and computing cost. In our dataset, the median episode length is 462 tokens. Setting the limit at 1,024 tokens keeps about 82% of episodes complete. We decided against a higher limit because it would include more episodes but would also need much more memory.

2.2 Stage 2: Value-weighted turn-level SFT

Stage 2 converts successful trajectories into local state–action examples. For each assistant turn, we create:

> **prompt** = dialogue/game history before the assistant action
> **completion** = next assistant action.

A successful transcript becomes many small local decisions, and the model is trained on the inference-time problem directly (given the current interaction state, produce the next legal assistant action). Decision-level factorisation of complex action spaces has recently been studied and applied for LLM game agents (Xu et al., 2025); in our setting, the "moves" are themselves conversational contributions.

Each row receives a lightweight heuristic value computed from the completion text. Clean, concise, non-leaky completions receive higher weight: v = 0.70
+ 0.15 $\cdot 1[1 \leq len(c) \leq 400]$
$+ 0.05 \cdot 1[nostop - tokenleakage]$
$+ 0.05 \cdot 1[norole - markerleakage]$
$+ 0.05 \cdot 1[len(c) > 0]$. The value is capped at 1.0. Because all source episodes are successful, most values concentrate around 1.00 for clean short completions and 0.85 for clean long completions. Weighting imitation by a scalar per-decision value follows reward- and advantage-weighted regression (Peters and Schaal, 2007; Wang et al., 2018; Peng et al., 2019), which LLM post-training instantiates as reward-filtered or reward-ranked fine-tuning (Gulcehre et al., 2023; Dong et al., 2023). Unlike equilibrium-derived Q-values (Xu et al., 2025) or learned advantages, our values are deliberately cheap textual proxies computed from the completion alone, prioritising clean, concise, non-leaky local decisions under a small compute budget.

We train with completion-only weighted SFT. Let ℓ_i be the mean token loss over the completion of example i, and let w_i be its value. The batch loss is:

$$L = \frac{\sum_i w_i \ell_i}{\sum_i w_i}.$$

We also train a uniform ablation where all turn-level weights are set to 1.0. This tests whether the

gain comes from factorisation alone or from value weighting.

2.3 Stage 3: Teacher-guided repair as constrained teacher repair

Stage 3 keeps the Stage 2 recipe but changes the data (we add a small amount of teacher-guided material while keeping Stage 2 rows dominant). A stronger LLM (Gemma-4-31B) teacher judges selected Stage 2 rows and generates invalid outputs for repair training, a minimal computational analogue of scaffolded, guided participation, in which a more capable partner corrects form and curates practice without supplying the learner's moves (Vygotsky, 1978; Bruner, 1983).

The teacher does not invent a new gold action, the target action always remains the original gold action from a successful Playpen trace, and invalid outputs appear only on the input side of repair examples.

For judged rows, the teacher receives the dialogue/game history before the action, the original gold assistant answer, the previous Stage 2 weight, and the strict Playpen output contract. It returns JSON scores such as `format_score`, `action_score`, `task_progress_score`, `done_score`, `overall_score`, a keep decision, a short reason, and invalid `bad_outputs`. The teacher prompt explicitly states that the clean target must remain the provided gold answer.

For kept rows, the teacher's overall score s mildly adjusts the original value v:

$$v' = \mathrm{clip}(v(0.75 + 0.50s),\ 0.40,\ 1.20).$$

The teacher can change how much an example counts, but it never changes what the model is trained to produce. In the final mixture, each judged row takes the place of its original in the Stage 2 pool rather than being added alongside it. The teacher therefore only reweights and filters existing data; no example is counted twice.

For repair examples, the teacher writes realistic near-misses of a gold action: added explanation, a missing `GO:` prefix, wrong casing, leaked role markers, stray punctuation, several actions at once, or a premature `DONE`. Each repair prompt shows the game history and one such invalid answer and asks the model to rewrite it correctly; the training target is always the original gold action. Producing small corruptions of a given answer is an easy task for an instruction-tuned model, so there is little room for it to invent content. As a check, we manually inspected 20% of the generated repairs and found no confabulated text.

The final submitted mixture is shown in Table 1.

Component	Count	Share
Original Stage 2 weighted-turn rows	30,000	96.5%
Teacher-judged/reweighted rows	1,000	3.2%
Teacher-generated repair rows	100	0.3%
Total	31,100	100%

Table 1: Final Stage 3 training mixture.

The teacher ratio is intentionally small. Development runs showed that heavier replay or repair can improve Playpen-specific clemscore while reducing statscore, suggesting over-specialisation.

2.4 Model adaptation, prompting, and post-processing

All stages use Qwen3.5-2B with LoRA adapter training (Hu et al., 2022) followed by merging into the base model. We use a standard configuration (rank 16, $\alpha = 32$, i.e. $\alpha = 2r$, dropout 0.05) and adapt all linear layers (`target_modules="all-linear"`), following evidence that adapting all layers rather than attention projections alone is important for matching full fine-tuning quality (Dettmers et al., 2023). A single adapter is created in Stage 1 and continued in all later weighted-SFT runs. The final submitted checkpoint is a full merged Hugging Face model, not only a LoRA adapter.

Teacher-guided augmentation uses two prompt types (Details in the Appendix). The judging prompt asks the teacher to return JSON-only scores, a keep/drop decision, a reason, and invalid outputs, while explicitly forbidding new gold answers. The repair prompt provides the game history and an invalid answer, and asks the model to output only the corrected valid answer. The teacher (Gemma-4-31B) is queried at near-deterministic temperature (0.1) for judging, and is drawn from a different model family than the student, avoiding the self-preference effects reported for same-family LLM judges. The full prompt templates are included in Appendix.

For Playpen evaluation, the model is registered with the local Hugging Face backend, the premade chat template, left padding, and `eos_to_cull=<\|im_end\|>`. After merging, `generation_config.json` is patched

to `eos_token_id=[248046, 248044]` and `pad_token_id=248044`, so the model stops at Qwen's chat end token as well as the base EOS.

3 Experimental Setup

3.1 Data

All training supervision derives from the `colab-potsdam/playpen-data` interactions train split. Validation data is used only for public evaluation and model selection, not for constructing training examples. No private evaluation data is used.

Training data comes in four forms: (1) successful full trajectories for Stage 1; (2) factorised turn-level state-action rows for Stage 2; (3) teacher-judged/reweighted rows for Stage 3; and (4) teacher-generated repair rows for Stage 3.

The full Stage 2 set contains approximately 106k factorised turn rows. The final Stage 3 mixture contains 31,100 rows: 30,000 original Stage 2 rows, 1,000 teacher-judged rows, and 100 repair rows. The mixture builder uses master seed 28, with derived draw seeds 29, 30, and 31 for the Stage 2, judged, and repair samples, respectively. Trainer seed is 42.

The dataset-building scripts reject paths containing validation/test/eval markers such as `validation`, `/val`, `test`, `playpen-eval`, `eval_results`, `leaderboard`, or `public`.

3.2 Training hyperparameters and compute

Setting	Stage 1	Stage 2	Final
Initialisation	Base + new LoRA	Stage 1 adapter	Stage 2 adapter
Data	Success dialogues	∼106k turn rows	31,100-row mix
Loss tokens	All	Completion-only	Completion-only
Example weights	None	Heuristic	Heur./teacher/repair
Learning rate	2e-4	5e-5	5e-5
Epochs	1.0	0.25	0.25
Effective batch	8	16	8–16

Table 2: Per-stage training configuration.

Stage 1 uses a learning rate of 2e-4, consistently for LoRA fine-tuning (Dettmers et al., 2023); Stage 2 and the final run reduce this to 5e-5 to limit drift when continuing an already-trained adapter. Fractional epochs are grounded in our own ablation: 0.50 epochs of Stage 2 training did not improve over 0.25 (Table 5), so all later runs use 0.25.

Training used NVIDIA A100 80GB GPUs: Stage 1 on a single process (effective batch size 8), Stage 2 with 2-GPU DDP (effective batch size 16), and the final run with runtime GPU of batch size 16.

All remaining heuristic constants, the value-formula terms (§2.2), the teacher-modulation coefficients and clip range, the repair weight of 0.75, and the judge-score threshold of 0.70, were set once by inspection under a fixed compute budget and were not tuned. Our ablations therefore test each mechanism as a whole rather than its constants.

3.3 Evaluation

We evaluate with the public Playpen validation pipeline using:

```
playpen eval <model> --suite all
```

The evaluation includes an interactive dialogue-game suite and a static suite. We report clemscore for interactive performance and statscore as a robustness check against over-specialisation. All evaluations use Playpen/clembench default decoding.

For runs where detailed breakdowns are available, clemscore decomposes as:

$$\text{clemscore} = \frac{\text{Avg.\%Played} \times \text{Avg. Quality}}{100}.$$

This separates two failure modes. % Played measures whether the model can validly participate without aborting. Quality measures how well the model performs once it is playing.

The official Qwen3.5-2B base scores are 13.05 clemscore and 44.02 statscore. Our local base run gives the 11.91, with 28.57 average % Played and 41.70 average Quality. We use the official base for headline improvement claims and the local base for playability/quality analysis.

4 Results and Analysis

4.1 Main results

Model / Method	Avg % Played	Avg Quality	Clemscore	Statscore
Base Qwen3.5-2B official	-	-	13.05	44.02
Base Qwen3.5-2B local	28.57	41.70	11.91	-
Stage 1: Success SFT	85.60	53.46	45.76	39.74
Stage 2: Uniform weights	-	-	46.18	40.70
Stage 2: Weighted-turn	83.77	63.65	53.32	41.61
Stage 2: Weighted-turn (0.5 epoch)	82.34	64.45	53.07	40.62
Final submitted model	83.30	68.57	57.12	42.68

Table 3: Main results across training stages.

The final submitted model improves clemscore by +44.07 over the official base model. It also improves over the strongest non-teacher baseline, Stage 2 Weighted-turn, by +3.80 clemscore and +1.07 statscore.

4.2 Playability and quality improve at different stages

The central pattern is simple, each stage improves a different aspect of interaction.

Transition	Δ Avg % Played	Δ Avg Quality
Base local → Stage 1	+57.03	+11.76
Stage 1 → Stage 2 Weighted	-1.83	+10.19
Stage 2 Weighted → Final	-0.47	+4.92

Table 4: Playability and quality deltas across stage transitions.

Stage 1 mostly improves playability. The local base model plays only 28.57% of episodes, while Stage 1 plays 85.60%. This suggests that full successful trajectories teach the interactional grammar of Playpen: action format, turn structure, and stopping behaviour.

Stage 2 operates at nearly constant playability but raises average Quality from 53.46 to 63.65. This supports the hypothesis that factorising trajectories into state-action examples exposes the local decision signal more directly than full-dialogue imitation.

The final teacher-guided stage further raises average Quality to 68.57 while maintaining similar playability. This suggests that the combined teacher-guided stage improves decision precision and strict-format behaviour rather than merely increasing the number of valid episodes.

Overall, the pipeline has a clear division of labour: complete trajectories make the model playable; value-weighted turns make it better; teacher-guided repair makes it more precise.

4.3 Ablations and negative results

We compare against four development variants: FAIPD-RR, a heavier replay-and-repair mixture trained after Stage 2; RR-scaled merge, which interpolates the replay-repair LoRA delta before merging; FAIPD-Q, which assigns Q-like heuristic weights using both successful and failed/aborted trajectories; and HEM-Mix, which performs loss-based hard-example mining over Stage 2 rows.

The ablations compare the final lightweight teacher-guided mixture against heavier alternatives: replay-and-repair training, scaled replay-repair merging, Q-like outcome weighting, and loss-based hard-example mining.

These results show that stronger intervention was not automatically better. FAIPD-RR improved clemscore over Stage 2 but reduced statscore, suggesting over-specialisation to benchmark-specific output behaviour. FAIPD-Q likely introduced noisy credit assignment by weighting failed and aborted trajectories. HEM-Mix may have selected high-loss examples that were difficult because of length or context rather than because they were useful training frontiers. The final method works because it preserves the successful Stage 2 distribution and adds only a small corrective signal from the teacher.

Method	Clemscore	Statscore	Interpretation
Stage 2 Weighted-turn	53.32	41.61	Strong non-teacher baseline
Stage 2 Weighted-turn, 0.50 epoch	53.07	40.62	More training did not help
FAIPD-RR	54.86	38.66	Replay/repair improved clem but hurt stat
RR-scaled merge, $\alpha = 0.75$	53.12	,	Scaling RR delta did not recover the gain
FAIPD-Q	51.06	40.84	Q-like heuristic weighting was noisy
HEM-Mix	51.41	40.60	Loss-based hard mining underperformed
Final submitted model	57.12	42.68	Best balance

Table 5: Ablation results and interpretation.

5 Submitted Model and Reproducibility

The exact public model URL and team identifier are anonymised in this review version to comply with ARR-style anonymous review. The checkpoint was provided through the official LM Playschool Challenge submission process and will be restored in the camera-ready version. The submitted system is a full merged 2B Hugging Face checkpoint evaluated with the local Hugging Face backend. No deviations from the public Playpen evaluation setup were made, except for standard local model registration, left padding, `eos_to_cull=<\|im_end\|>`, and the post-merge generation-config patch described above.

6 Limitations

The submitted checkpoint was selected using public validation results, so the public score may overestimate expected private-test performance. The method does not perform online reinforcement learning, explicit search, or planning. The teacher does not propose new strategies; it only judges and repairs existing successful-trace actions. Wall-clock runtime is not included here but will be included in the final submission. The robustness of using an LLM as a teacher is not explored.

References

Anne Beyer, Kranti Chalamalasetti, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. 2024. clembench-2024: A challenging, dynamic, complementary, multilingual benchmark and underlying flexible framework for LLMs as multi-action agents. *arXiv preprint arXiv:2405.20859*.

Jerome S. Bruner. 1983. *Child's Talk: Learning to Use Language*. W. W. Norton, New York.

Kranti Chalamalasetti, Jana Götze, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. 2023. clembench: Using game play to evaluate chat-optimized language models as conversational agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun Shum, and Tong Zhang. 2023. RAFT: Reward rAnked FineTuning for generative foundation model alignment. *Transactions on Machine Learning Research*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*.

Nicola Horst, Davide Mazzaccara, Antonia Schmidt, Michael Sullivan, Filippo Momentè, Luca Franceschetti, Philipp Sadler, Sherzod Hakimov, Alberto Testoni, Raffaella Bernardi, Raquel Fernández, Alexander Koller, Oliver Lemon, David Schlangen, Mario Giulianelli, and Alessandro Suglia. 2025. Playpen: An environment for exploring learning from dialogue game feedback. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 29854–29891, Suzhou, China. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*.

Jan Peters and Stefan Schaal. 2007. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 745–750.

David Schlangen, Sherzod Hakimov, Kranti Chalamalasetti, Jonathan Jordan, and Philipp Sadler. 2025. A third paradigm for LLM evaluation: Dialogue game-based evaluation using clembench. *arXiv preprint arXiv:2507.08491*.

Lev S. Vygotsky. 1978. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press.

Qing Wang, Jiechao Xiong, Lei Han, Peng Sun, Han Liu, and Tong Zhang. 2018. Exponentially weighted imitation learning for batched historical data. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pages 6291–6300.

Kaixuan Xu, Jiajun Chai, Sicheng Li, Yuqian Fu, Yuanheng Zhu, and Dongbin Zhao. 2025. DipLLM: Finetuning LLM for strategic decision-making in diplomacy. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

A Appendix

A.1 Judging prompt

Each selected Stage 2 row is sent to the teacher as a two-message conversation. The system message is:

> You are a strict JSON−only data−quality judge and synthetic−error generator for a dialogue−game agent. Return one valid JSON object only. Do not use markdown. Do not explain outside JSON.

The user message is the following template:

> You are judging one training example for a dialogue−game agent.
>
> The agent must obey a strict output contract:
> – Output exactly one final answer.
> – For movement tasks, valid movement answers look like exactly:
> GO: north
> GO: south
> GO: east
> GO: west
> – If the task is complete, the answer should be exactly:
> DONE
> – No explanation.
> – No markdown.
> – No role prefix.
> – No punctuation around the answer.
> – No stop tokens.
> – No multiple actions.

```
You are given:
1. The dialogue state/history BEFORE the assistant answer.
2. The original gold assistant answer from a successful or
      high−quality training trace.
3. The old Stage 2 training weight.

You must NOT invent a new gold answer.
The clean target must remain the provided gold answer.

Your tasks:
A. Score how good this training row is for teaching strict,
      correct dialogue−game behaviour.
B. Generate realistic bad assistant outputs that a smaller
      model might produce for this exact example.
   These bad outputs should be close to the gold answer but
      invalid or less strict.

Examples of badness:
− extra explanation
− missing GO: prefix
− wrong casing
− punctuation
− markdown
− role leakage
− stop−token leakage
− multiple actions
− premature DONE
− verbose DONE

Return exactly one JSON object and nothing else.

Required JSON schema:
{
  "format_score": 1.0,
  "action_score": 1.0,
  "task_progress_score": 1.0,
  "done_score": 1.0,
  "overall_score": 1.0,
  "keep": true,
  "reason": "short reason",
  "bad_outputs": ["bad output 1", "bad output 2"]
}

Rules:
− Scores must be floats from 0.0 to 1.0.
− "overall_score" should summarize whether the row is
      useful for training.
− "keep" should be false only if the row looks malformed,
      useless, contradictory, or too ambiguous.
− "bad_outputs" must contain at most {max_bad_outputs}
      strings.
− "bad_outputs" must NOT include the exact clean target.
− "bad_outputs" must NOT be empty unless keep is false.

Row index:
{row_index}

Old Stage 2 weight:
{original_weight}

Dialogue state/history before answer:
{context}

Gold assistant answer:
{target}
```

Placeholder	Replaced with
`{context}`	The message history before the assistant action, serialised as JSON and truncated to its final 7,000 characters (the most recent game state matters most for the next action).
`{target}`	The original gold assistant answer from the Stage 2 row.
`{original_weight}`	The row's existing Stage 2 heuristic value.
`{row_index}`	The row's index in the Stage 2 file, used for resumable processing and provenance tracking.
`{max_bad_outputs}`	4 in all our runs.

Table 6: Template placeholders in the judging prompt.

A.2 Repair prompt

Each kept `bad_output` produces one repair training example. The prompt is a single user message built from the following template; the supervised target is always the original gold action, never the teacher's text:

```
You are playing a dialogue game. The previous assistant
      answer was invalid because it did not follow the exact
      required output format or it contained extra text.

Game state/history before the assistant answer:
{context}

Invalid assistant answer:
{bad}

Rewrite the assistant answer correctly. Output only one
      final valid answer and nothing else.
```