CS 329Z:Engineering AI Agents

LLM-as-Judge & Evaluation Infrastructure

Diyi Yang, Michael Ryan, John Yang

Roadmap

LLM-as-Judge & Evaluation Infrastructure

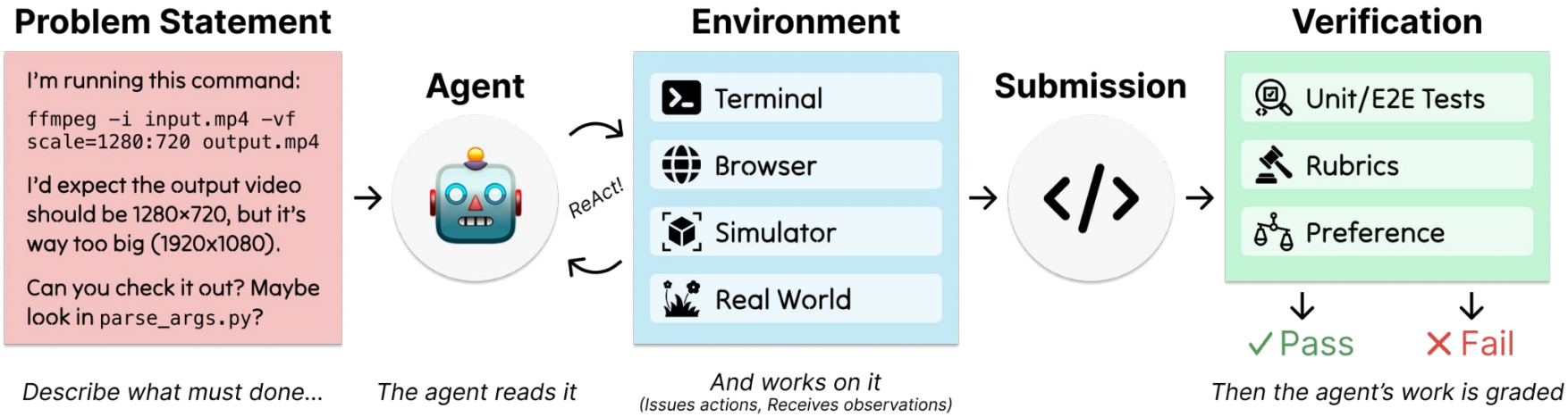
- Motivation [7 min]
- Grader Types [13 min]
- LM-as-a-Judge [35 min]
- Biases + Validate Judgment [20 min]
- Handle Nondeterminism [5 min]

tl;dr: Last lecture we discussed the high level concepts and best practices around agent benchmarks. This lecture we'll zoom in on the nitty gritty of verification: general approaches, their mechanics, and how to ensure reliability + objectivity.

Motivation

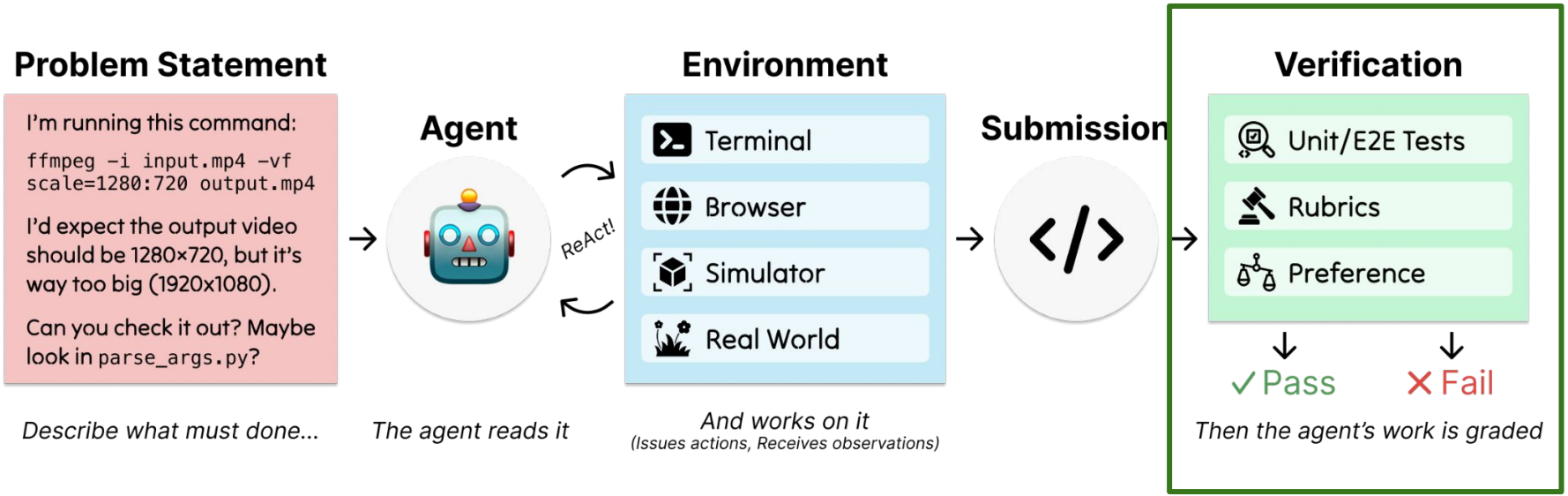
Motivation

Recall from last lecture, the 3 components of a benchmark's **task formulation**



Motivation

Recall from last lecture, the 3 components of a benchmark's **task formulation**



Motivation

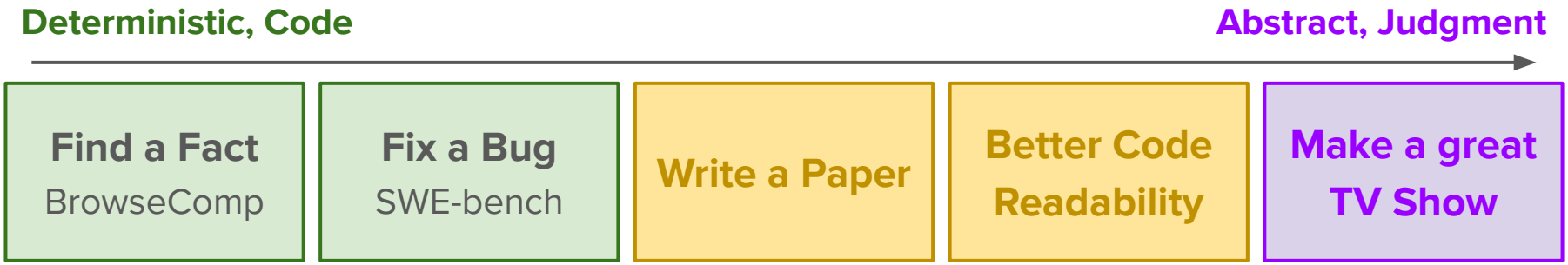
What different kinds of verification techniques are there? One popular technique is to encode expected behavior as deterministic checks in the form of **unit tests**.

Find a Fact	**Fix a Bug**	**Place an Order**
BrowseComp	SWE-bench	Tau-bench
`answer == gold`	`pytest all pass`	`order in orders.db?`
Check w/ code	Check w/ code	Check w/ code

But is checking with code all we need?

Motivation

But many agent tasks don't have deterministic, singular solutions.



Unit tests might not cover...

- 2+ valid answers, beyond reference
- Process, beyond the outcome
- Quality, beyond correctness
- Partial credit, beyond pass/fail

Motivation

A miscalibrated grader may also artificially deflate or inflate benchmark scores.

		According to grader: **Correct**	According to grader: **Incorrect**
In reality, solution is	**Correct**	**True Positive**	**False Negative**: On τ^2-bench, Claude Opus 4.5 found a better solution that grader did not account for. [A\ Blog]
	Incorrect	**False Positive**: In SWE-bench, some patches pass F2P tests, but fail unregistered P2P tests [Wang 2025]	**True Negative**

Motivation

Agents are non-deterministic. Across multiple attempts on the same task, some attempts may fail while others will succeed, how do we account for this variance?

Let's say you run 5 rollouts / trials of an agent on a task, and get: ❌ ✅ ❌ ❌ ✅

How do you report this outcome?

- **Average (40%)**: Communicates expected success rate per try.
- **At least once (Yes)**: Says whether agent is capable of solving task *at all*.
- **Solved every time (No)**: Says whether you can count on agent

Motivation

Let's say you run a task 5 times, and get: ❌✅❌❌✅

How do you report this outcome?

- **Average (40%)**: Communicates expected success rate per try.
  ⇒ *Hides*: So is each attempt a coin flip? Or tasks are always all pass/fail?
- **At least once (Yes)**: Says whether agent is capable of solving task *at all*.
  ⇒ *Hides*: Luck + cost of retrying. What if you scale arbitrarily to k retries?
- **Solved every time (No)**: Says whether you can count on agent
  ⇒ *Hides*: Near-misses and potential infra failings. Goes towards 0% w/ more k.

Motivation

Each section of today's lecture addresses several key questions centered around how to set up reliable, fair verifiers for a benchmark / task:

- **Grader Types**: What kinds of graders exist? When do I use each?
- **LM-as-a-Judge**: When would I need LM-as-a-judge? How do I set it up?
- **Biases + Validating Judgment**: How can I make sure the verdict is trustworthy?
- **Handling Non-determinism**: There's variance across multiple runs. How do we account for it?

Grader Types

Grader Types

Depending on the task, different graders are more or less effective at **analyzing task artifacts meaningfully** across a range of cost considerations.

	Cost	Speed	Reliable	Flexibility	Scale	Nuance
Code	Cheap	Fast	Yes	Low	High	Low
Human	Expensive	Slow	Partly	High	Low	High
LM Judge	Moderate	Fast	Partly	High	High	High

Each of these graders can look at two things:

Final Environment State

Trajectory

Code Based Graders

Deterministic checks encoded as runnable scripts that operate on env. or transcript.

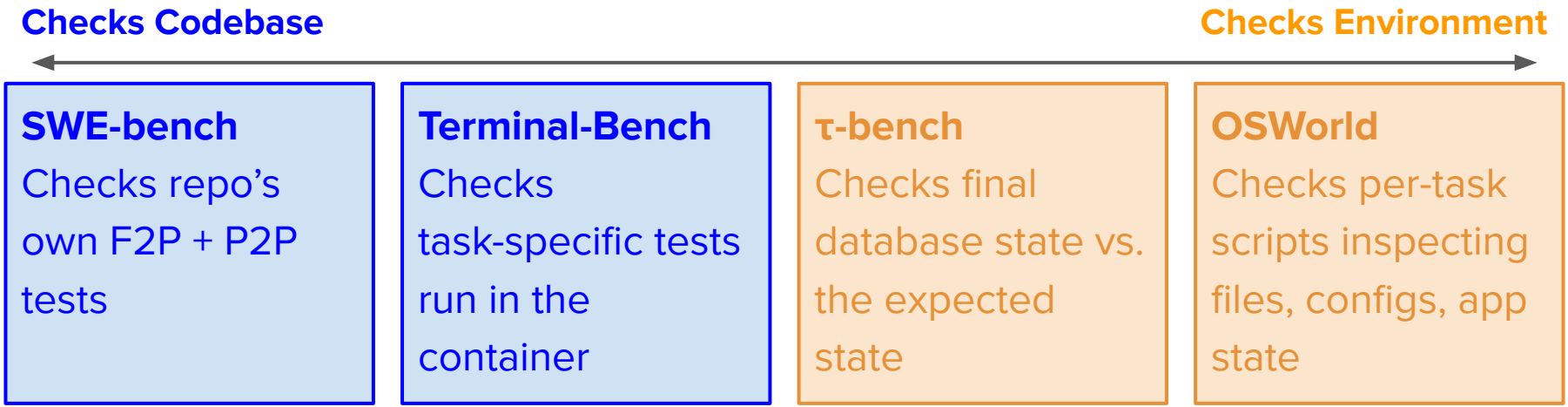
Common Checks

- String / regex match
- Unit tests (F2P, P2P)
- Environment state checks
- Static analysis
- Tool-call checks (was X called?)

```
def grade(env, transcript):
  # outcome: did the fix work?
  ok = run('pytest tests/') == 0
  # outcome: is the order there?
  ok &= env.db.has_order('u42')
  # transcript: identity checked?
  ok &= 'verify_id' in transcript.tools
  return ok
```

Code Based Graders

Different tasks' unit tests check different components of a rollout on a task.



Code Based Graders

But code-oriented grading is not perfect! They capture exactly what they capture, and nothing more.

Prescriptive
If a function or class is checked, the solution *must* have that function or class.

Static
Tests don't evolve over time. Once they're written, they are "set in stone".

Reductive
Anything that the tests don't cover will implicitly get registered as a "pass".

Human Graders

People judge the work directly, can be effective for more open-ended work.

Common Setups

- Expert review (domain specialists)
- Crowd votes (many non-experts)
- Pairwise preference: A vs. B
- Live use: real users work with the agent, then rate it



Human Graders

Different evaluations, depending on their topic, will consult different demographics.

Chatbot Arena

Who: anonymous crowd

How: vote between two anonymous models' replies

Score: votes aggregated into Elo ratings

GDPval

Who: occupational experts across 44 occupations

How: blind comparison of model vs. human-expert deliverables

Score: win rate vs. experts

Co-Gym (Real)

Who: real users with their own tasks

How: collaborate with the agent in a shared workspace, then rate it

Score: outcome + collaboration process

Human Graders

Human voting is a gold standard (high fidelity to reality), but collection takes time.

Expensive

Every new model needs a new round of raters.

Days or weeks per evaluation

Disagreement

Two raters can reach different verdicts.

Report inter-annotator agreement

Hard to Rerun

Rater pools and contexts change; results don’t reproduce exactly.

Sessions can’t replay

LM-as-a-Judge Graders

Use an LM as a stand in for a human grader! Prompt the LM with a rubric.

Common Checks

- Reference-based: Does it match the gold answer?
- Rubric scoring: Rate 1-10
- Pairwise: Which is better?
- Assertions: is this statement about the output true?

Judge prompt

Given the [question], the agent's [response], and the [correct answer], decide only whether the final answer matches.

Judge output

```
extracted_answer: …
reasoning: …
correct: yes
```

LM-as-a-Judge Graders

Benchmarks differ in how much judgment they ask of the model.

Specific: Does it match? ⟷ **Open-ended Preferences**

BrowseComp / HLE

Judge sees
Response + gold answer

Decides
Is the final answer equivalent?

MT-bench

Judge sees
Multi-turn conversation

Decides
score 1-10, or pick the better of two

AlpacaEval

Judge sees
Model's vs. reference's reply

Decides
which is better (length-controlled win rate)

LM-as-a-Judge Graders

Scales like code, judges like human, but inherits problems of both.

Non-deterministic

The same input can get different verdicts.

Run multiple judges or samples

Biased

Favors certain positions, longer answers, its own style.

How to deal with bias?

Needs Calibration

Only as good as its agreement with humans.

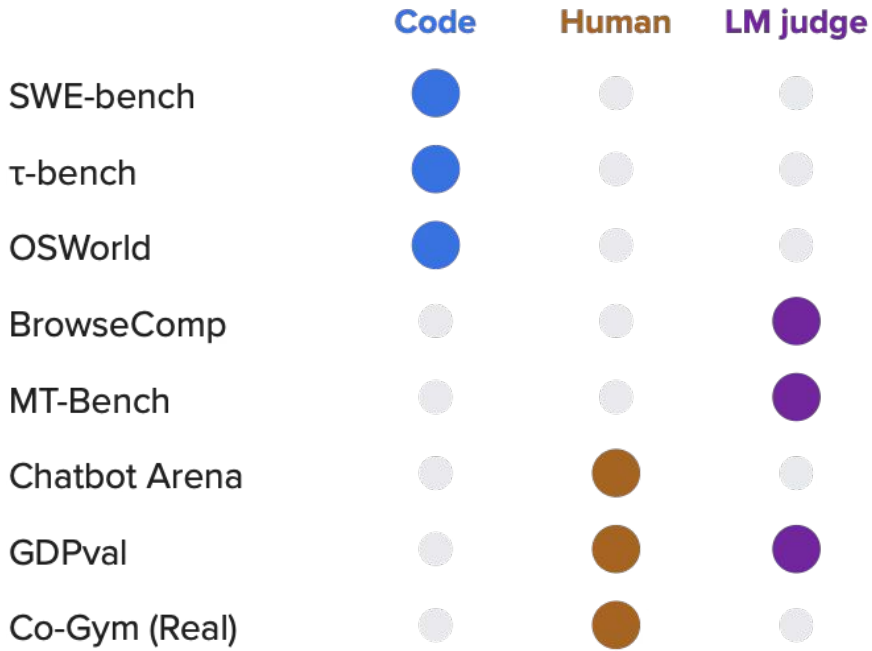
We'll explore how to align judgment.

Summary

Evals sometimes use combinations of graders: deterministic, code-based unit tests where possible; LM where necessary; humans judiciously.

Multiple ways you can aggregate eval results together:

- **All must pass**: tests pass and the LM rubric says the code is clean
- **Weighted**: scores add up to a threshold
- **Calibrate**: humans spot-check LM judgment to maintain agreement



LM-as-a-Judge

LM-as-a-Judge

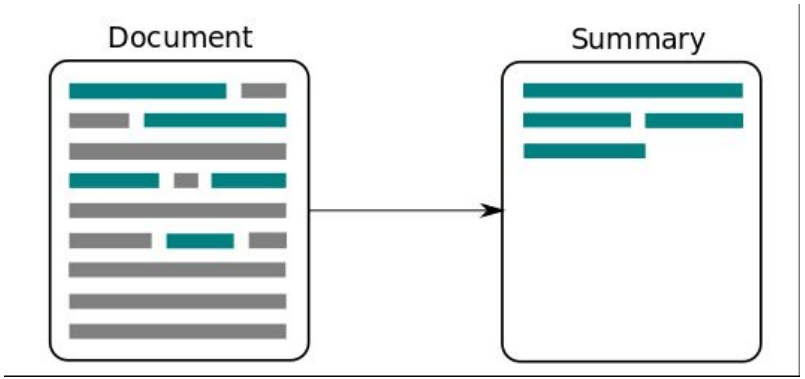
Let’s review the emergence and development of LM-as-a-judge grading.

- **[Phase 1] NLP Metrics**: Can LMs supplant metrics for traditional NLP tasks (e.g., dialogue generation, text summarization)
- **[Phase 2] Human Preferences**: Can LMs decide which completion is better in a manner that’s faithful to human inclinations?
- **[Phase 3] Human Actions**: Can LMs interact with digital or real worlds the way we humans can, and make meaningful judgments based on such interactions?

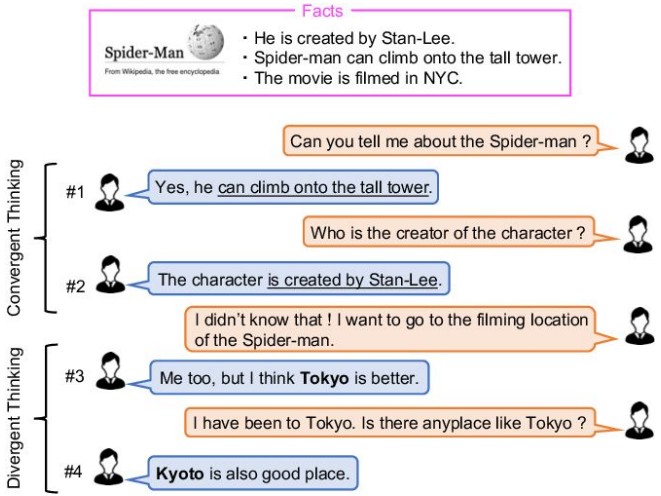
[Phase 1] NLP Metrics

G-Eval

Motivation: Can an LLM evaluate open-ended text better than n-gram overlap?



Text Summarization

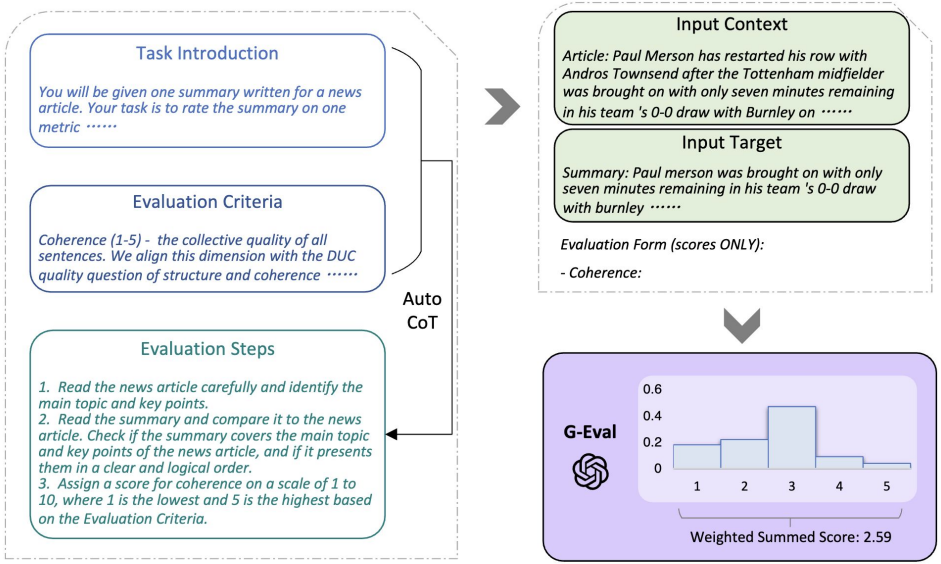


Dialogue Generation

Liu et al., "G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment," EMNLP 2023

G-Eval

Method: Give GPT-4 the criteria (e.g., “coherence”), have it first write out how it will judge, then ask for a 1–5 score. Score = average over likelihood of 1, 2, 3, 4, 5.



You will be given one summary written for a news article. Your task is to rate the summary on one metric.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

Given news article + AI summary, GPT-4 assigns 1 - 5 score for factual consistency.

G-Eval

Findings: 0.514 Spearman with humans on summarization, beating prior metrics by a large margin. Early warning: it may favor LLM-written text.

Metrics	Coherence		Consistency		Fluency		Relevance		AVG	
	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ
ROUGE-1	0.167	0.126	0.160	0.130	0.115	0.094	0.326	0.252	0.192	0.150
ROUGE-2	0.184	0.139	0.187	0.155	0.159	0.128	0.290	0.219	0.205	0.161
ROUGE-L	0.128	0.099	0.115	0.092	0.105	0.084	0.311	0.237	0.165	0.128
BERTScore	0.284	0.211	0.110	0.090	0.193	0.158	0.312	0.243	0.225	0.175
MOVERSscore	0.159	0.118	0.157	0.127	0.129	0.105	0.318	0.244	0.191	0.148
BARTScore	0.448	0.342	0.382	0.315	0.356	0.292	0.356	0.273	0.385	0.305
UniEval	0.575	0.442	0.446	0.371	0.449	0.371	0.426	0.325	0.474	0.377
GPTScore	0.434	–	0.449	–	0.403	–	0.381	–	0.417	–
G-EVAL-3.5	0.440	0.335	0.386	0.318	0.424	0.347	0.385	0.293	0.401	0.320
- Probs	0.359	*0.313*	0.361	*0.344*	0.339	*0.323*	0.327	*0.288*	0.346	*0.317*
G-EVAL-4	**0.582**	**0.457**	**0.507**	**0.425**	**0.455**	**0.378**	**0.547**	**0.433**	**0.514**	**0.418**
- Probs	0.560	*0.472*	0.501	*0.459*	0.438	*0.408*	0.511	*0.444*	0.502	*0.446*
- CoT	0.564	0.454	0.493	0.413	0.403	0.334	0.538	0.427	0.500	0.407

Table 1: Summary-level Spearman (ρ) and Kendall-Tau (τ) correlations of different metrics on SummEval benchmark. G-EVAL without probabilities (*italicized*) should not be considered as a fair comparison to other metrics on τ, as it leads to many ties in the scores. This results in a higher Kendall-Tau correlation, but it does not fairly reflect the true evaluation ability. More details are in Section 4.

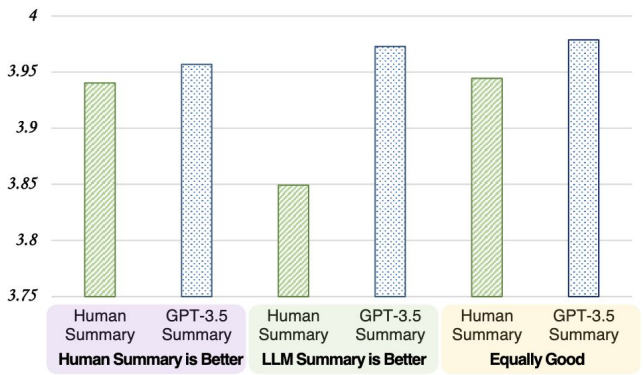


Figure 2: Averaged G-EVAL-4's scores for human-written summaries and GPT-3.5 summaries, divided by human judges' preference.

GPTScore

Motivation: Can a generative LM score any aspect of text, given only an NL description of that aspect?

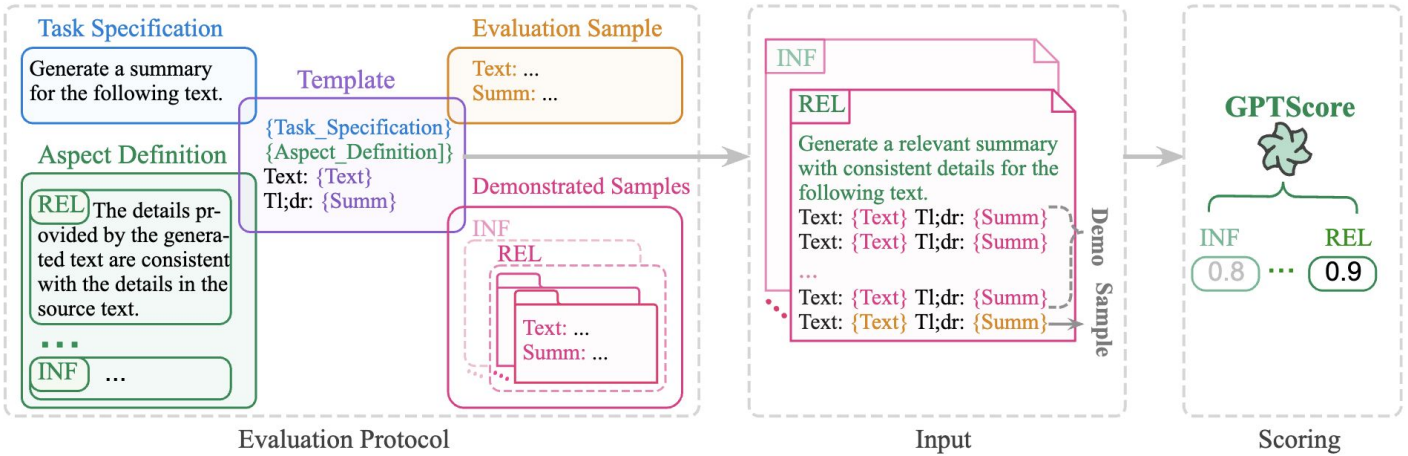


Figure 2. The framework of GPTSCORE. We include two evaluation aspects *relevance (REL)* and *informative (INF)* in this figure and use the evaluation of *relevance (REL)* of the text summarization task to exemplify our framework.

Fu et al., "GPTScore: Evaluate as You Desire," NAACL 2024 (arXiv Feb 2023)

GPTScore

Method: Prompt LM with instruction describing the aspect (e.g., “write a summary of…”), then score candidate text by how probable the LM finds it as a continuation.

Aspect	Task	Definition
Semantic Coverage (COV)	Summ	How many semantic content units from the reference text are covered by the generated text?
Factuality (FAC)	Summ	Does the generated text preserve the factual statements of the source text?
Consistency (CON)	Summ, Diag	Is the generated text consistent in the information it provides?
Informativeness (INF)	Summ, D2T, Diag	How well does the generated text capture the key ideas of its source text?
Coherence (COH)	Summ, Diag	How much does the generated text make sense?
Relevance (REL)	Diag, Summ, D2T	How well is the generated text relevant to its source text?
Fluency (FLU)	Diag, Summ, D2T, MT	Is the generated text well-written and grammatical?
Accuracy (ACC)	MT	Are there inaccuracies, missing, or unfactual content in the generated text?
Multidimensional Quality Metrics (MQM)	MT	How is the overall quality of the generated text?
Interest (INT)	Diag	Is the generated text interesting?
Engagement (ENG)	Diag	Is the generated text engaging?
Specific (SPE)	Diag	Is the generated text generic or specific to the source text?
Correctness (COR)	Diag	Is the generated text correct or was there a misunderstanding of the source text?
Semantically appropriate (SEM)	Diag	Is the generated text semantically appropriate?
Understandability (UND)	Diag	Is the generated text understandable?
Error Recovery (ERR)	Diag	Is the system able to recover from errors that it makes?
Diversity (DIV)	Diag	Is there diversity in the system responses?
Depth (DEP)	Diag	Does the system discuss topics in depth?
Likeability (LIK)	Diag	Does the system display a likeable personality?
Flexibility (FLE)	Diag	Is the system flexible and adaptable to the user and their interests?
Inquisitiveness (INQ)	Diag	Is the system inquisitive throughout the conversation?

GPTScore

Findings: Tested 19 LMs (80M–175B) on 4 tasks, 22 aspects, 37 datasets: describing the task + aspect in the instruction makes scores more reliable.

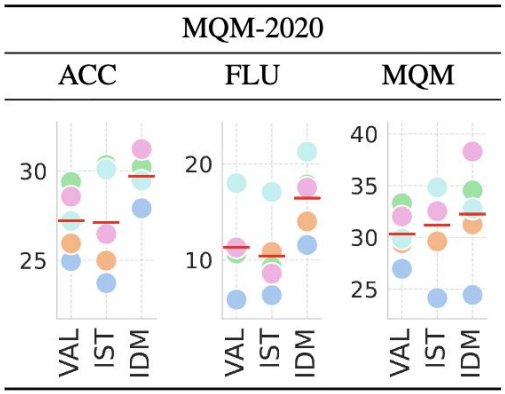


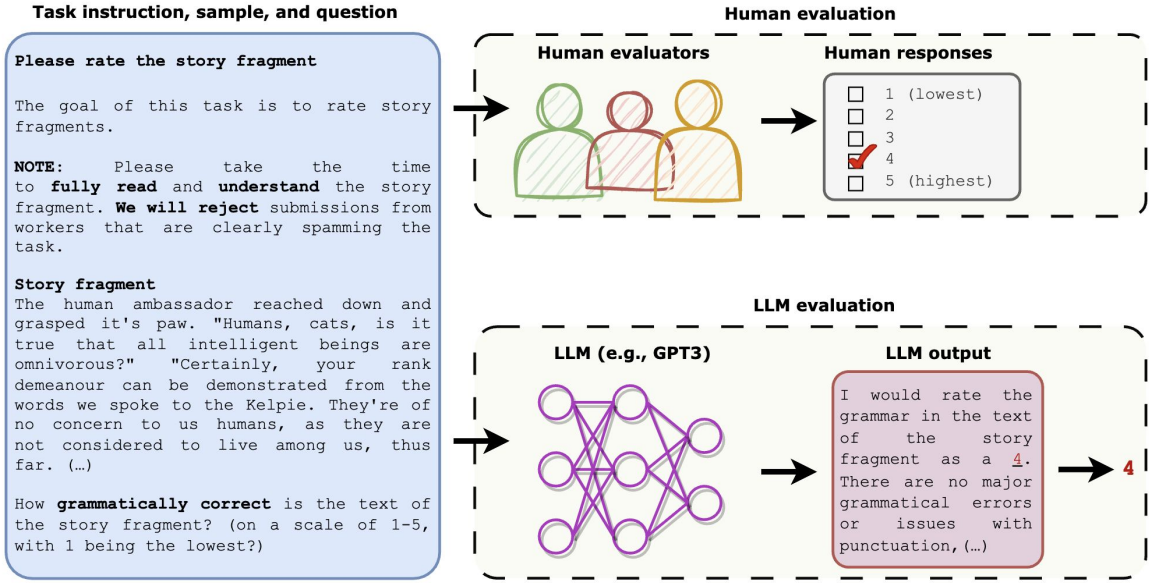
Figure 4. Experimental results for GPT3-based variants in the machine translation task. Here, blue, orange, green, pink, and cyan dot denote that GPTSCORE is built based on a01 (●), b01 (●), c01 (●), d01 (●), and d03 (●), respectively. The red lines (—) denote the average performance of GPT3-based variants.

Model	INF			NAT			FLU		
	VAL	IST	IDM	VAL	IST	IDM	VAL	IST	IDM
BAGEL									
GPT3	35.4	38.3†	43.6†,‡	21.7	26.5†	36.9†,‡	30.5	32.9†	43.4†,‡
GPT2	40.8	43.2†	40.2	31.4	33.0†	33.5†,‡	36.7	39.3†	41.3†,‡
OPT	38.7	39.3†	38.6	31.4	30.0	33.7†,‡	37.7	37.1†	41.5†,‡
FT5	41.5	41.5	39.1	26.5	29.7†	28.6†	38.1	41.1†	40.3†
Avg.	39.1	40.6†	40.3†	27.7	29.8†	33.2†,‡	35.8	37.6†	41.6†,‡
SFRES									
GPT3	30.4	25.1	31.5†,‡	25.0	30.4†	26.5†	31.2	30.9	26.1
GPT2	22.5	25.1†	20.5	31.0	31.9†	37.0†,‡	20.0	33.1†	36.2†,‡
OPT	25.2	26.9†	24.3	26.2	30.0†	36.6†,‡	21.3	25.6†	30.6†,‡
FT5	24.0	21.9	19.7	34.3	34.6†	36.8†,‡	22.0	17.8	19.7‡
Avg.	25.5	24.7	24.0	29.1	31.7†	34.2†,‡	23.6	26.8†	28.2†,‡

Table 5. The average of Spearman correlation the models based on GPT3, GPT2, OPT, and FT5 on BAGEL and SFRES datasets in data-to-text task.

Can LLMs Replace Human Evaluation?

Motivation: Human evaluation is hard to reproduce and unstable. Can an LLM fill out the exact questionnaire human raters get?



Chiang & Lee, “Can Large Language Models Be an Alternative to Human Evaluations?” ACL 2023

Can LLMs Replace Human Evaluation?

Method: Give LLMs the identical instructions, samples, and questions used with expert raters (English teachers), on story generation and adversarial attacks.

Open-ended story generation is a task to generate a short story based on a given prompt. We use the `WritingPrompts` dataset (Fan et al., 2018), which is composed of pairs of short prompts and human-written stories collected from the subreddit `WritingPrompts`. In the `WritingPrompts`, the users are given a short prompt, and they need to write a story based on the short prompt.[1]

In this experiment, we use LLM evaluation and human evaluation to rate the stories generated by humans and the stories generated by a story generation model.

We present the LLMs and the human evaluators with a short description, and the story to be evaluated, formatted as shown in Figure 1. Following Karpinska et al. (2021), we evaluate the stories on four different attributes. The four attributes and their corresponding questions are as follows:

1. *Grammaticality*: How **grammatically correct** is the text of the story fragment?
2. *Cohesiveness*: How well do **the sentences** in the story fragment **fit together**?
3. *Likability*: How **enjoyable** do you find the story fragment?
4. *Relevance*: Now read the PROMPT based on which the story fragment was written.
   Prompt: `[PROMPT]`.
   How **relevant** is the **story fragment** to the **prompt?**

Can LLMs Replace Human Evaluation?

Findings: LLMs rank texts the way experts do, and are stable across instruction wording and sampling. But they are more lenient than experts on unnatural text.

Evaluator	Grammaticality		Cohesiveness		Likability		Relevance	
	Mean_{STD}	$\text{IAA}_{\%}$	Mean_{STD}	$\text{IAA}_{\%}$	Mean_{STD}	$\text{IAA}_{\%}$	Mean_{STD}	$\text{IAA}_{\%}$
Human-written stories								
Human	$3.76_{0.95}$	$0.33_{20.5}$	$4.29_{0.82}$	0.32_{27}	$3.78_{1.10}$	$0.08_{9.5}$	$3.35_{1.48}$	0.05_{8}
T0	$2.55_{1.47}$	0.16_{10}	$2.98_{1.45}$	0.11_{4}	$3.18_{1.53}$	0.12_{7}	$2.93_{1.64}$	0.02_{6}
curie	$3.19_{0.47}$	$0.07_{46.5}$	$2.82_{0.46}$	$0.01_{47.5}$	$2.85_{0.37}$	$0.11_{0.65}$	$3.06_{0.40}$	$0.11_{0.64}$
davinci	$4.22_{0.38}$	0.26_{35}	$4.54_{0.47}$	$0.37_{39.5}$	$3.99_{0.38}$	$0.49_{68.5}$	$4.40_{0.79}$	$0.71_{48.5}$
ChatGPT	$3.83_{0.60}$		$3.55_{0.88}$		$2.44_{0.89}$		$3.29_{1.50}$	
GPT-2-generated stories								
Human	$3.56_{0.91}$	$0.10_{19.5}$	$3.19_{1.07}$	0.14_{17}	$2.59_{1.29}$	$-0.21_{3.5}$	$2.38_{1.40}$	$-0.03_{8.5}$
T0	$2.44_{1.49}$	0.05_{9}	$3.02_{1.51}$	0.07_{6}	$3.00_{1.59}$	0.16_{6}	$2.82_{1.61}$	0.04_{6}
curie	$3.23_{0.51}$	0.01_{38}	$2.82_{0.45}$	0.02_{50}	$2.86_{0.37}$	$0.09_{65.5}$	$3.01_{0.43}$	0.11_{61}
davinci	$4.07_{0.35}$	$0.35_{45.5}$	$4.26_{0.45}$	0.42_{42}	$3.84_{0.42}$	0.52_{62}	$4.02_{0.74}$	$0.69_{42.5}$
ChatGPT	$2.98_{0.76}$		$2.48_{0.71}$		$1.59_{0.67}$		$2.02_{1.21}$	

Table 1: LLM evaluation and human evaluation results of human-written stories and GPT-2-generated stories. For each evaluated attribute, we report its mean Likert scale and the standard deviation. We also report the inter-annotator agreement (IAA) among three annotators using Krippendorff's α. The subscript in the IAA column (%) is used to denote the percentage of the stories where all three annotators exactly agree on a rating.

[Phase 2] Human Preferences

[Phase 2] Human Preferences

Phase 1 asked whether LMs can replace text metrics. Phase 2 asks: **can an LM judge which output people would prefer**?

1. Can we learn a judge from human preferences?
2. Can a prompted LM stand in for human raters?
3. How do we make these judges better and trustworthy?

Tl;dr: the same preference judge can score a model (eval) or reward it (training). We'll cover both in the upcoming papers!

A Brief Primer

There has been a long history of developing neural methods that mimic humans.

Deep Reinforcement Learning from Human Preferences

Paul F Christiano
OpenAI
paul@openai.com

Jan Leike
DeepMind
leike@google.com

Tom B Brown
nottombrown@gmail.com

Miljan Martic
DeepMind
miljanm@google.com

Shane Legg
DeepMind
legg@google.com

Dario Amodei
OpenAI
damodei@openai.com

A long line of work studies reinforcement learning from human ratings or rankings, including Akrour et al. (2011), Pilarski et al. (2011), Akrour et al. (2012), Wilson et al. (2012), Sugiyama et al. (2012), Wirth and Fürnkranz (2013), Daniel et al. (2015), El Asri et al. (2016), Wang et al. (2016), and Wirth et al. (2016). Other lines of research considers the general problem of reinforcement learning from preferences rather than absolute reward values (Fürnkranz et al., 2012; Akrour et al., 2014), and optimizing using human preferences in settings other than reinforcement learning (Machwe and Parmee, 2006; Secretan et al., 2008; Brochu et al., 2010; Sørensen et al., 2016).

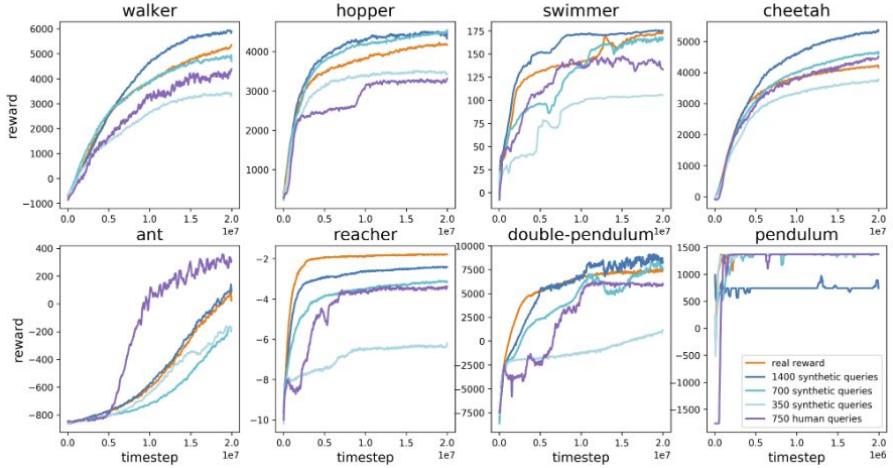
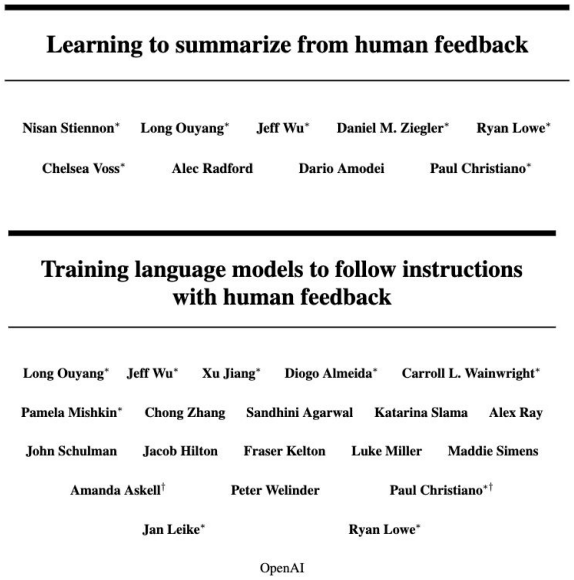


Figure 2: Results on MuJoCo simulated robotics as measured on the tasks' true reward. We compare our method using real human feedback (purple), our method using synthetic feedback provided by an oracle (shades of blue), and reinforcement learning using the true reward function (orange).

Christiano, Paul F., et al. "Deep reinforcement learning from human preferences." NeurIPS 2017.

Learning to summarize from human feedback

Motivation: ROUGE and reference summaries are rough proxies for quality. Can we train a model to predict what humans prefer, and optimize for that instead?



Learning to summarize from human feedback

Nisan Stiennon* Long Ouyang* Jeff Wu* Daniel M. Ziegler* Ryan Lowe*
Chelsea Voss* Alec Radford Dario Amodei Paul Christiano*

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
Amanda Askell† Peter Welinder Paul Christiano*†
Jan Leike* Ryan Lowe*
OpenAI

As language models become more powerful, training and evaluation are increasingly bottlenecked by the data and metrics used for a particular task. For example, summarization models are often trained to predict human reference summaries and evaluated using ROUGE, but both of these metrics are rough proxies for what we really care about—summary quality. In this work, we show that it is possible to significantly improve summary quality by training a model to optimize for human preferences. We collect a large, high-quality dataset of human comparisons between summaries, train a model to predict the human-preferred summary, and use that model as a reward function to fine-tune a summarization policy using reinforcement learning. We apply our method to a version of the TL;DR dataset of Reddit posts [63] and find that our models significantly outperform both human reference summaries and much larger models fine-tuned with supervised learning alone. Our models also transfer to CNN/DM news articles [22], producing summaries nearly as good as the human reference without any news-specific fine-tuning.2 We conduct extensive analyses to understand our human feedback dataset and fine-tuned models.3 We establish that our reward model generalizes to new datasets, and that optimizing our reward model results in better summaries than optimizing ROUGE according to humans. We hope the evidence from our paper motivates machine learning researchers to pay closer attention to how their training loss affects the model behavior they actually want.

Stiennon et al., “Learning to summarize from human feedback,” NeurIPS 2020

Learning to summarize from human feedback

Method: Collect human comparisons between summaries, train a reward model to predict the preferred one, then use it as the reward to fine-tune a summarizer with RL.

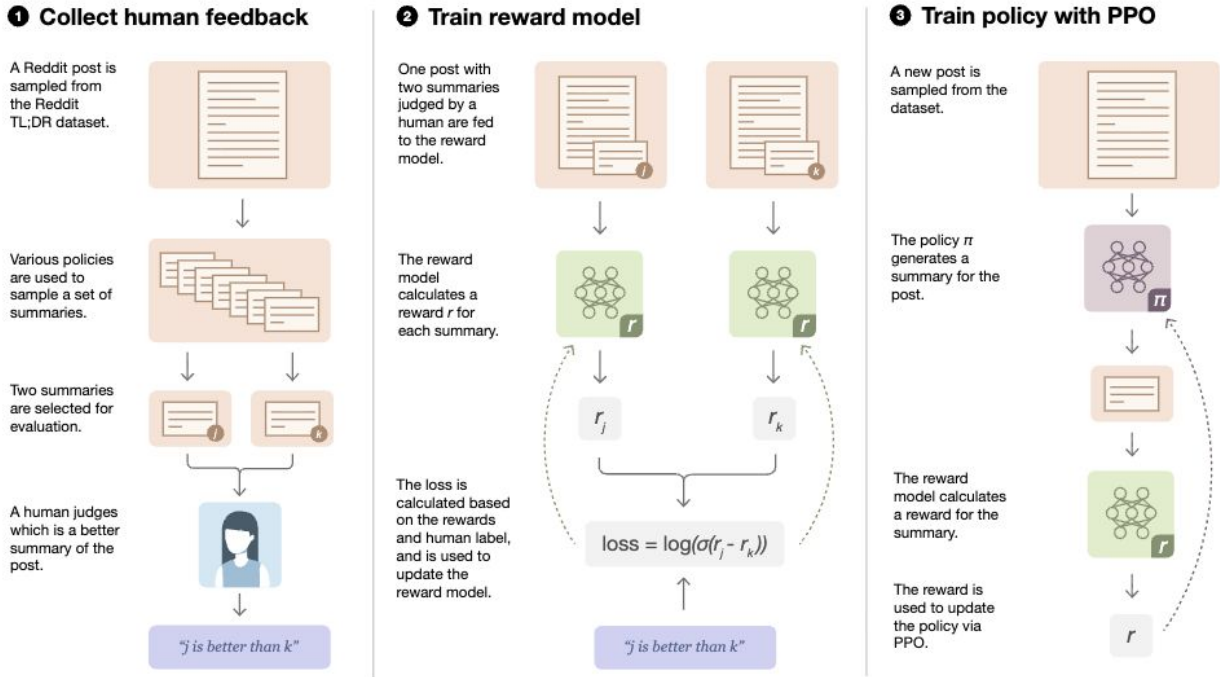


Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

Learning to summarize from human feedback

Findings: Summaries beat human references + larger supervised models; reward model predicts human preferences better than ROUGE and transfers to news.

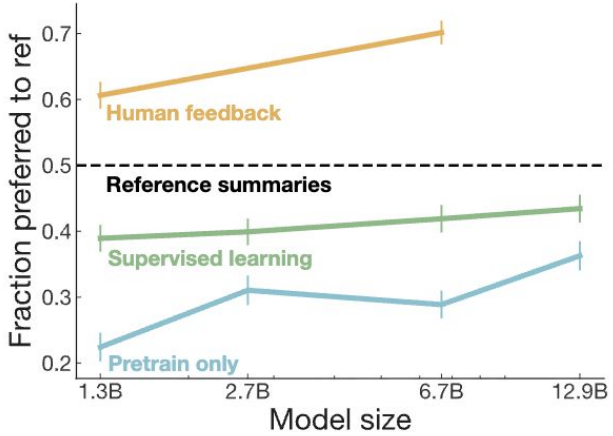


Figure 1: Fraction of the time humans prefer our models' summaries over the human-generated reference summaries on the TL;DR dataset.[4] Since quality judgments involve an arbitrary decision about how to trade off summary length vs. coverage within the 24-48 token limit, we also provide length-controlled graphs in Appendix F; length differences explain about a third of the gap between feedback and supervised learning at 6.7B.

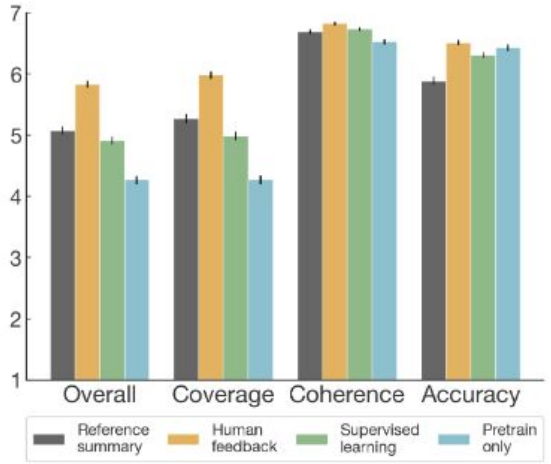


Figure 3: Evaluations of four axes of summary quality on the TL;DR dataset.

InstructGPT

Motivation: Generalizes the “Learning to Summarize…” strategy to beyond summarization.

Making language models bigger does not inherently make them better at following a user’s intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user… In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models InstructGPT. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters…

Table 1: Distribution of use case categories from our API prompt dataset.

Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

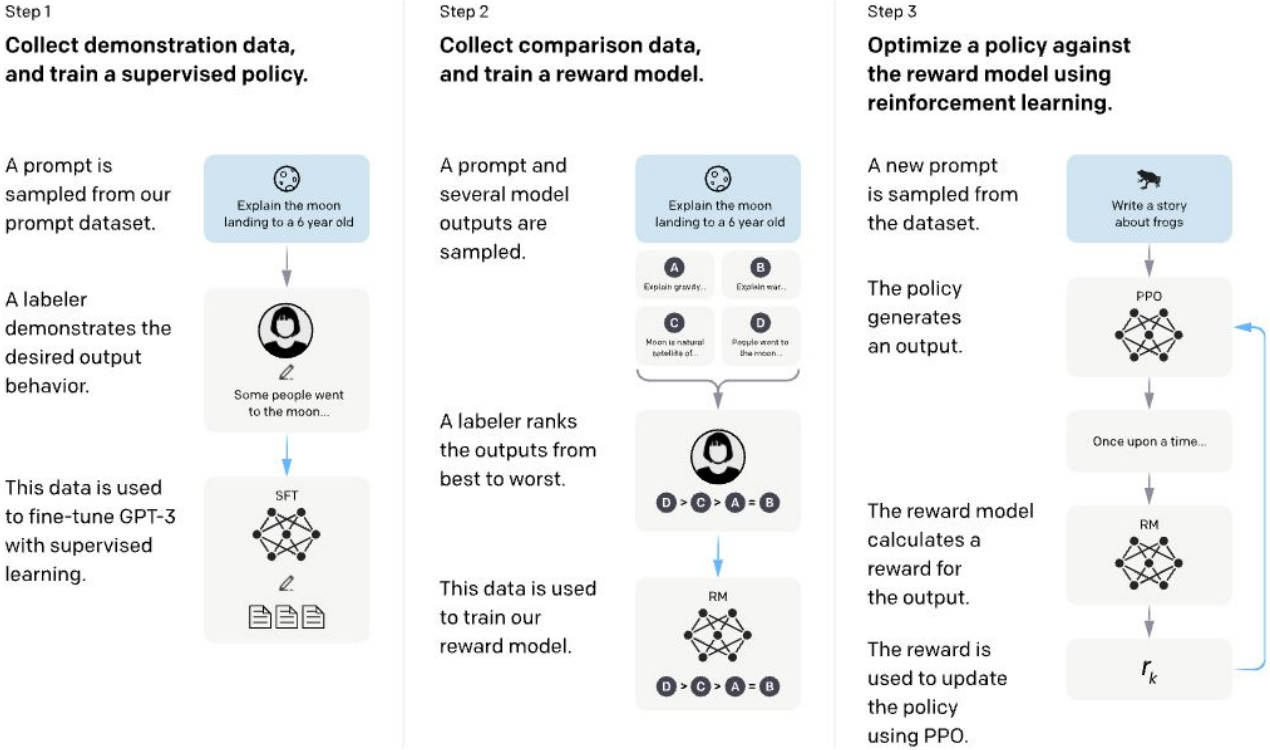
Table 2: Illustrative prompts from our API prompt dataset. These are fictional examples inspired by real usage—see more examples in Appendix A.2.1.

Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.
Rewrite	This is the summary of a Broadway play: """ {summary} """ This is the outline of the commercial for that play: """

Ouyang, Long, et al. "Training language models to follow instructions with human feedback." NeurIPS 2022

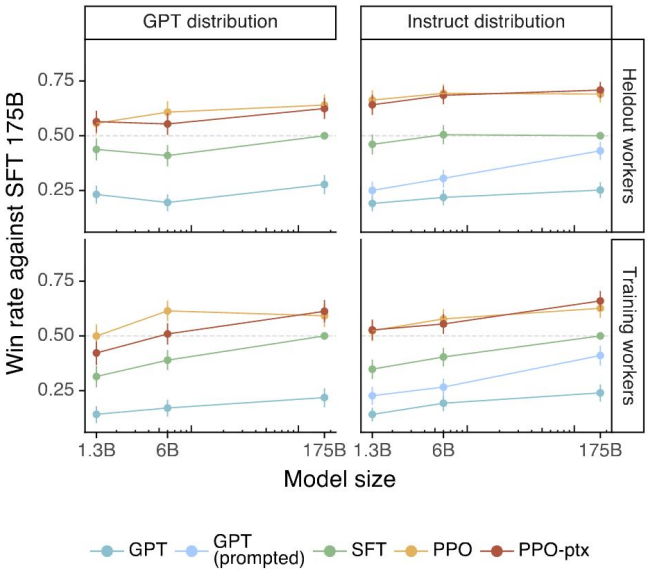
InstructGPT

Method: The 3 steps introduced in “Learning to Summarize” works in general!



InstructGPT

Findings: Labelers preferred outputs from the 1.3B InstructGPT over the 175B GPT-3, a model 100× larger. Also more truthful, less toxic, with minimal regressions.



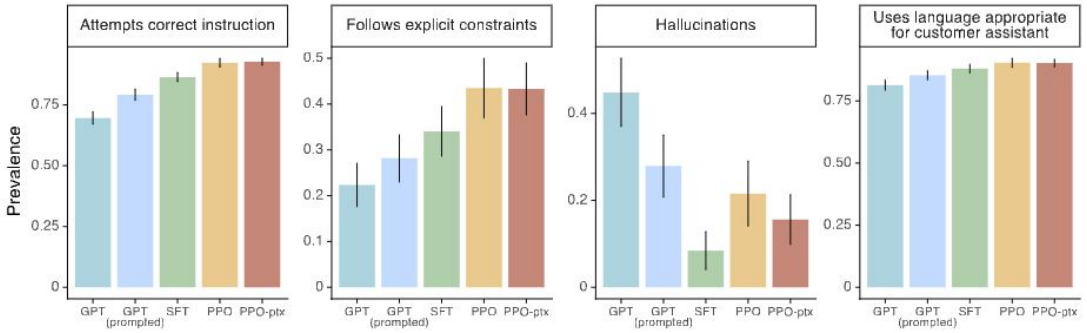


Figure 4: Metadata results on the API distribution. Note that, due to dataset sizes, these results are collapsed across model sizes. See Appendix E.2 for analysis that includes model size. Compared to GPT-3, the PPO models are more appropriate in the context of a customer assistant, are better at following explicit constraints in the instruction and attempting the correct instruction, and less likely to 'hallucinate' (meaning, making up information on closed domain tasks like summarization).

MT-Bench + Chatbot Arena

Motivation: Can an LM stand in for human preference when grading chatbots?

Because these models are often trained with RLHF, they already exhibit strong human alignment. We call this approach *"LLM-as-a-judge"*. This approach has been tried in our earlier blog post [8] and other concurrent or follow-up work [5, 29, 14, 12, 52, 18, 33, 40, 7, 43]. However, there has not been a systematic study of this approach.

Table 1: Sample multi-turn questions in MT-bench.

Category		Sample Questions
Writing	1st Turn	Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.
	2nd Turn	Rewrite your previous response. Start every sentence with the letter A.
Math	1st Turn	Given that $f(x) = 4x^3 - 9x - 14$, find the value of $f(2)$.
	2nd Turn	Find x such that $f(x) = 0$.
Knowledge	1st Turn	Provide insights into the correlation between economic indicators such as GDP, inflation, and unemployment rates. Explain how fiscal and monetary policies ...
	2nd Turn	Now, explain them again like I'm five.

Zheng et al., "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," NeurIPS 2023

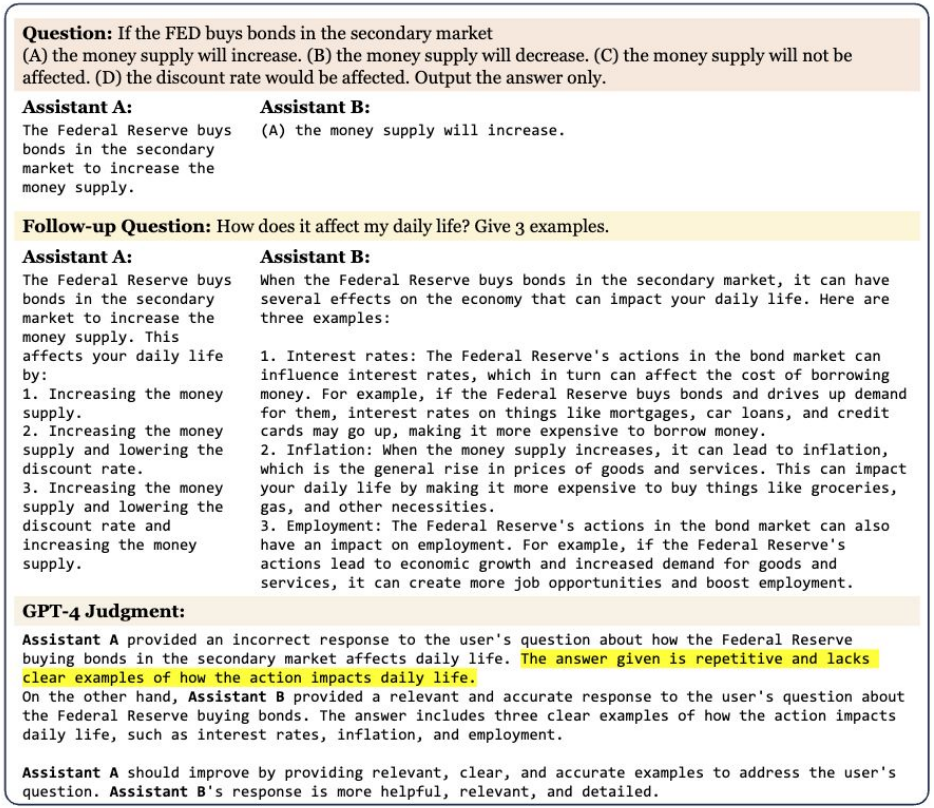
MT-Bench + Chatbot Arena

Method:

- Collect human preferences (80 multi-turn MT-Bench questions + Chatbot Arena votes)
- Then test 3 judge formats:

We propose 3 LLM-as-a-judge variations. They can be implemented independently or in combination:

- **Pairwise comparison**. An LLM judge is presented with a question and two answers, and tasked to determine which one is better or declare a tie. The prompt used is given in Figure 5 (Appendix).
- **Single answer grading**. Alternatively, an LLM judge is asked to directly assign a score to a single answer. The prompt used for this scenario is in Figure 6 (Appendix).
- **Reference-guided grading**. In certain cases, it may be beneficial to provide a reference solution if applicable. An example prompt we use for grading math problems is in Figure 8 (Appendix).



Question: If the FED buys bonds in the secondary market
(A) the money supply will increase. (B) the money supply will decrease. (C) the money supply will not be affected. (D) the discount rate would be affected. Output the answer only.

Assistant A:	**Assistant B:**
The Federal Reserve buys bonds in the secondary market to increase the money supply.	(A) the money supply will increase.

Follow-up Question: How does it affect my daily life? Give 3 examples.

Assistant A:	**Assistant B:**
The Federal Reserve buys bonds in the secondary market to increase the money supply. This affects your daily life by: 1. Increasing the money supply. 2. Increasing the money supply and lowering the discount rate. 3. Increasing the money supply and lowering the discount rate and increasing the money supply.	When the Federal Reserve buys bonds in the secondary market, it can have several effects on the economy that can impact your daily life. Here are three examples: 1. Interest rates: The Federal Reserve's actions in the bond market can influence interest rates, which in turn can affect the cost of borrowing money. For example, if the Federal Reserve buys bonds and drives up demand for them, interest rates on things like mortgages, car loans, and credit cards may go up, making it more expensive to borrow money. 2. Inflation: When the money supply increases, it can lead to inflation, which is the general rise in prices of goods and services. This can impact your daily life by making it more expensive to buy things like groceries, gas, and other necessities. 3. Employment: The Federal Reserve's actions in the bond market can also have an impact on employment. For example, if the Federal Reserve's actions lead to economic growth and increased demand for goods and services, it can create more job opportunities and boost employment.

GPT-4 Judgment:

Assistant A provided an incorrect response to the user's question about how the Federal Reserve buying bonds in the secondary market affects daily life. The answer given is repetitive and lacks clear examples of how the action impacts daily life.
On the other hand, **Assistant B** provided a relevant and accurate response to the user's question about the Federal Reserve buying bonds. The answer includes three clear examples of how the action impacts daily life, such as interest rates, inflation, and employment.

Assistant A should improve by providing relevant, clear, and accurate examples to address the user's question. **Assistant B**'s response is more helpful, relevant, and detailed.

Figure 1: Multi-turn dialogues between a user and two AI assistants—LLaMA-13B (Assistant A) and Vicuna-13B (Assistant B)—initiated by a question from the MMLU benchmark and a follow-up instruction. GPT-4 is then presented with the context to determine which assistant answers better.

MT-Bench + Chatbot Arena

Findings: GPT-4 agrees with humans over 80% of the time, the same rate humans agree with each other. But it shows position, verbosity, self-enhancement biases.

Table 2: Position bias of different LLM judges. Consistency is the percentage of cases where a judge gives consistent results when swapping the order of two assistants. "Biased toward first" is the percentage of cases when a judge favors the first answer. "Error" indicates wrong output formats. The two largest numbers in each column are in bold.

Judge	Prompt	Consistency	Biased toward first	Biased toward second	Error
Claude-v1	default	23.8%	**75.0%**	0.0%	1.2%
	rename	56.2%	11.2%	**28.7%**	**3.8%**
GPT-3.5	default	46.2%	**50.0%**	1.2%	2.5%
	rename	51.2%	38.8%	6.2%	**3.8%**
GPT-4	default	**65.0%**	30.0%	5.0%	0.0%
	rename	**66.2%**	28.7%	5.0%	0.0%

MT-Bench + Chatbot Arena

Findings: GPT-4 agrees with humans over 80% of the time, the same rate humans agree with each other. But it shows position, verbosity, self-enhancement biases.

Table 7: Category-wise win rate of models.

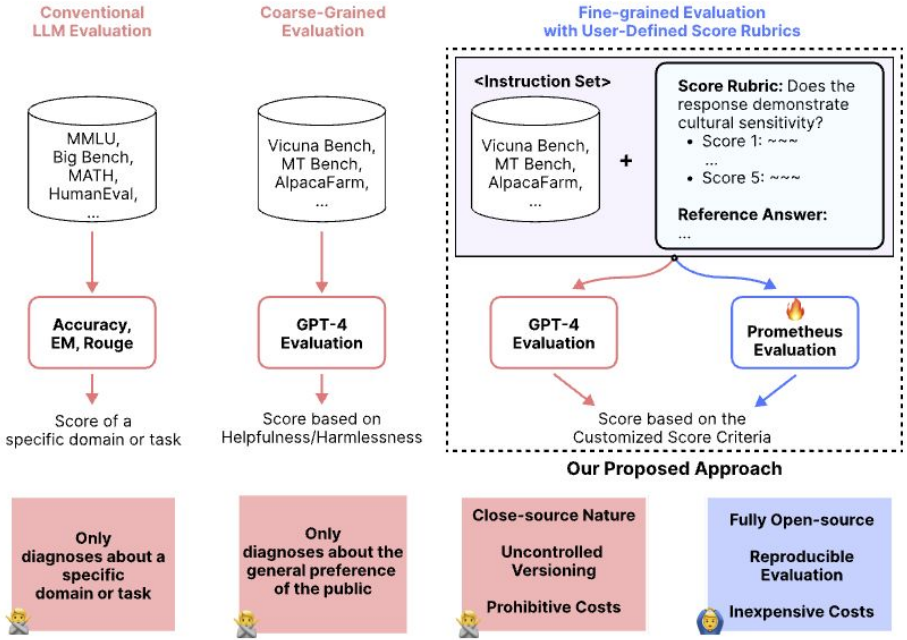
Model	Writing	Roleplay	Reasoning	Math	Coding	Extraction	STEM	Humanities
GPT-4	61.2%	67.9%	49.3%	66.1%	56.3%	66.2%	76.6%	72.2%
GPT-3.5	50.9%	60.6%	32.6%	63.8%	55.0%	48.8%	52.8%	53.8%
Vicuna-13B	39.7%	39.2%	20.1%	18.0%	36.9%	29.2%	47.0%	47.5%
LLaMA-13B	15.1%	15.1%	7.8%	7.5%	2.1%	9.3%	6.8%	10.1%

Prometheus

Motivation: Can you train an OSS model to judge with rubrics? (instead of GPT-4)

Recently, using a powerful proprietary Large Language Model (LLM) (e.g., GPT-4) as an evaluator for long-form responses has become the de facto standard. However, for practitioners with large-scale evaluation tasks and custom criteria in consideration (e.g., child-readability), using proprietary LLMs as an evaluator is unreliable due to the closed-source nature, uncontrolled versioning, and prohibitive costs.

In this work, we propose PROMETHEUS, a fully open-source LLM that is on par with GPT-4's evaluation capabilities when the appropriate reference materials (reference answer, score rubric) are accompanied. We first construct the FEEDBACK COLLECTION, a new dataset that consists of 1K fine-grained score rubrics, 20K instructions, and 100K responses and language feedback generated by GPT-4. Using the FEEDBACK COLLECTION, we train PROMETHEUS, a 13B evaluator LLM that can assess any given long-form text based on *customized* score rubric provided by the user.



Kim et al., "Prometheus: Inducing Fine-grained Evaluation Capability in Language Models," ICLR 2024

Prometheus

Method: Fine-tune Llama-2-Chat-13B on the Feedback Collection (1K rubrics, 20K instructions, 100K GPT-4 feedback); input includes a rubric and a reference answer.

Taking these into consideration, we construct each instance within the FEEDBACK COLLECTION to encompass four components for the *input* (instruction, response to evaluate, customized score rubric, reference answer) and two components in the *output* (feedback, score). An example of an instance is shown in Figure 2 and the number of each component is shown in Table 1.

The four components for the input are as follows:

1. **Instruction**: An instruction that a user would prompt to an arbitrary LLM.
2. **Response to Evaluate**: A response to the instruction that the evaluator LM has to evaluate.
3. **Customized Score Rubric**: A specification of novel criteria decided by the user. The evaluator should focus on this aspect during evaluation. The rubric consists of (1) a description of the criteria and (2) a description of each scoring decision (1 to 5).
4. **Reference Answer**: A reference answer that would receive a score of 5. Instead of requiring the evaluator LM to solve the instruction, it enables the evaluator to use the mutual information between the reference answer and the response to make a scoring decision.

The two components for the output are as follows:

1. **Feedback**: A rationale of why the provided response would receive a particular score. This is analogous to Chain-of-Thoughts (CoT), making the evaluation process interpretable.
2. **Score**: An integer score for the provided response that ranges from 1 to 5.

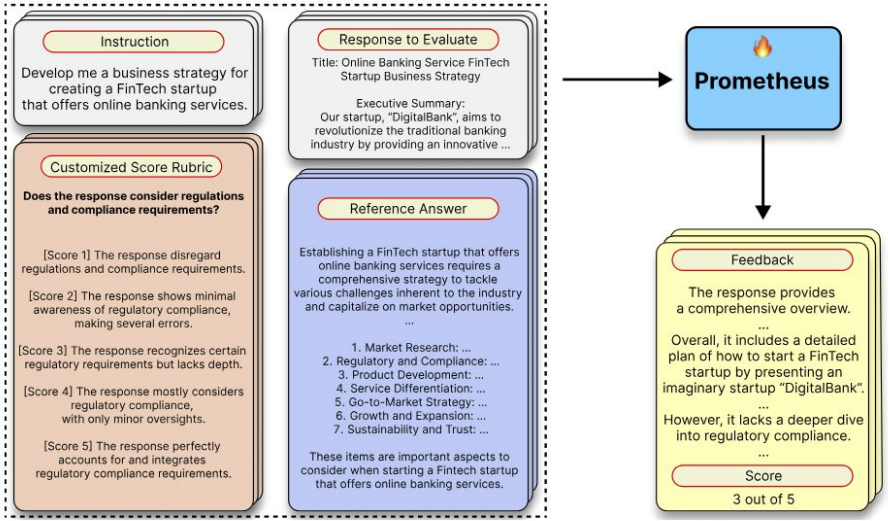
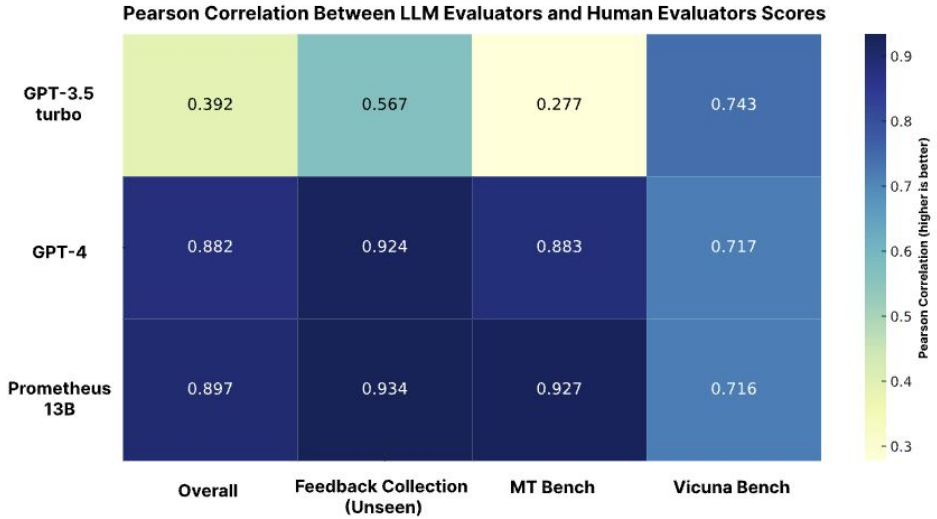
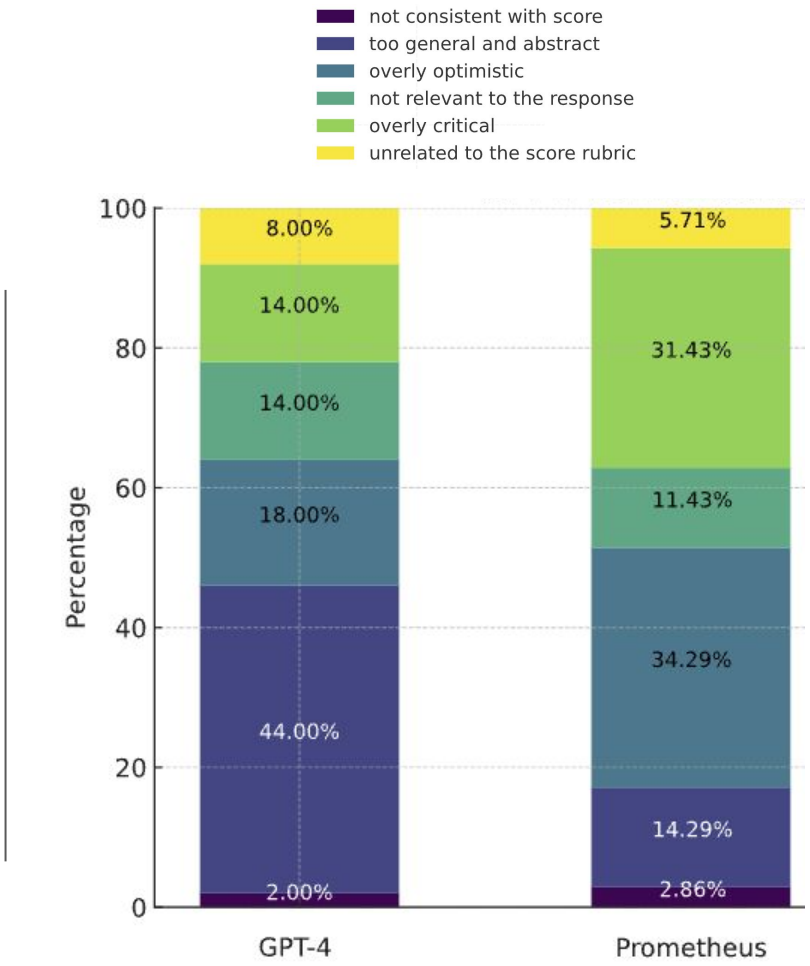


Figure 2: The individual components of the FEEDBACK COLLECTION. By adding the appropriate reference materials (Score Rubric and Reference Answer) and training on GPT-4's feedback, we show that we could obtain a strong open-source evaluator LM.

Prometheus

Findings: 0.897 Pearson with humans on 45 custom rubrics, on par with GPT-4 (0.882) and far above GPT-3.5 (0.392).





Briefly, what do rubrics look like? [Pointwise vs. Pairwise]

Two ways to ask a judge: “how good is this?” or “which is better?”

Prompt: Explain why the sky is blue to a 10-year-old.

Response A: “Sunlight hits air molecules, and shorter wavelengths undergo stronger Rayleigh scattering, roughly $\propto 1/\lambda^4$.”

Response B: “Sunlight has every color mixed in. Air bounces blue light around the most, so blue reaches your eyes from all over the sky.”

Pointwise: Rate each answer on scale



Pairwise: Compare, which is better?



[Phase 2] Human Preferences

Can an LM judge which output people would prefer?

1. Can we learn a judge from human preferences?

   Learning from summarizing, InstructGPT

2. Can a prompted LM stand in for human raters?

   MT-bench & ChatbotArena

3. How do we make these judges better and trustworthy?

   Prometheus

[Phase 3] Human Actions

[Phase 3] Human Actions

Phase 2 is very single-turn / dialogue centric. Phase 3 asks: **can an LM judge an agent's actions and their effects, not just its words?**

- Can an LM tell whether an agent's trajectory succeeded?
- Can the judge act like an agent, inspecting the environment itself?
- How reliable are these judges compared with human experts?

Tl;dr: Let's try judging actions, environments, and trajectories (beyond messages). Like Phas 2, the same judge can score an agent (eval) or reward it (training)

WebVoyager

Motivation: Live websites don't have an answer key or database to check. Can a VLM look at what an agent did and judge whether it succeeded?

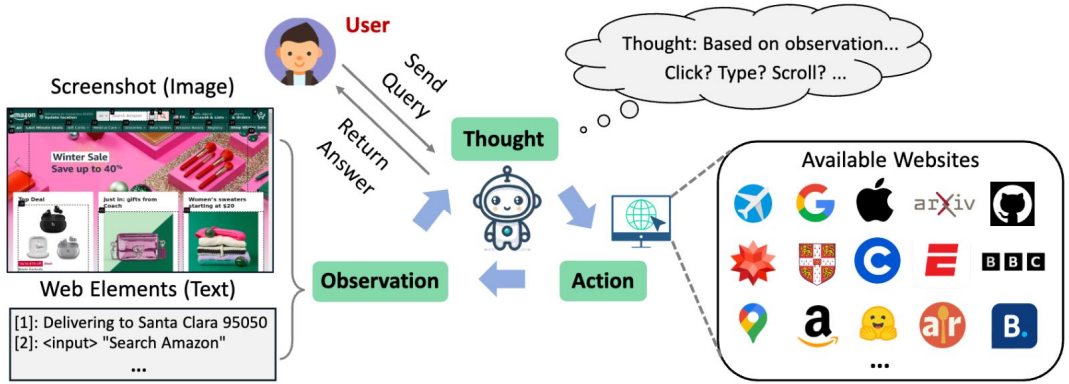
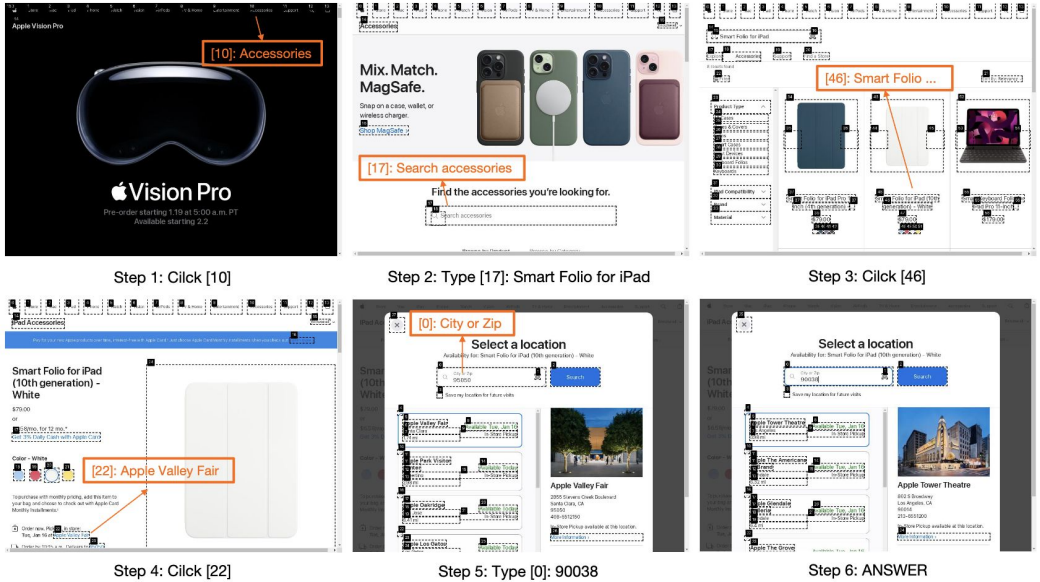


Figure 1: The overall workflow of WebVoyager. WebVoyager takes web tasks assigned by a human and automatically browses the web online. At each step, WebVoyager selects actions based on screenshots and text (the 'type' of the web element and its contents). Once the task is completed, the answers will be returned to the user. For example, for a user query: "Find the cost of a 2-year protection for PS4 on Amazon.", the agent interacts with Amazon online, locates the PS4, identifies the 2-year protection price, and returns "$30.99" to the user.

He et al., "WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models," ACL 2024

WebVoyager

Method: Save screenshots of agent's navigation, then have GPT-4V judge the trajectory and final answer; humans grade the same tasks to check the judge.



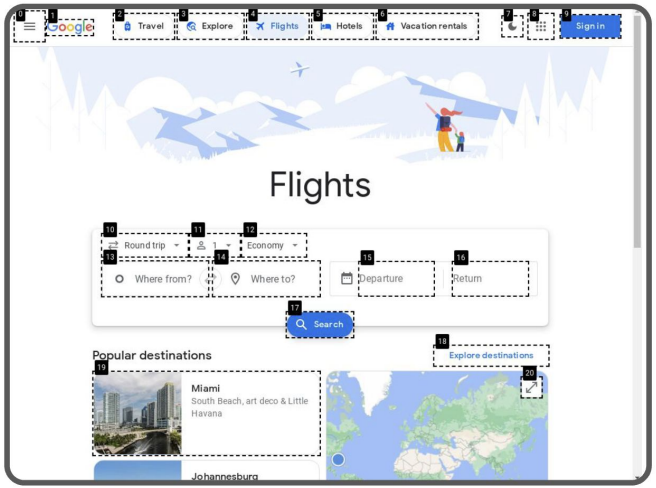


Figure 2: Examples of webpage screenshots provided to the agent. We add borders to most of the interactive elements on the web pages and label them with numerical tags in the top left corner.

WebVoyager

Findings: WebVoyager reaches 59.1% success on live websites (vs. 40.1% text-only), and the GPT-4V judge's scores track human labels closely overall (57.1%), though individual sites can differ by 10+ points.

	Allrecipes	Amazon	Apple	ArXiv	GitHub	Booking	ESPN	Coursera
GPT-4 (All Tools)	11.1%	17.1%	44.2%	14.0%	48.8%	22.7%	31.8%	31.0%
WebVoyager$_{\text{Text-only}}$	**55.6%**	31.7%	34.9%	32.6%	61.0%	2.3%	36.4%	23.8%
WebVoyager	53.3%	**58.5%**	**65.1%**	**51.2%**	**63.4%**	**43.2%**	**38.6%**	**73.8%**
WebVoyager$_{\text{Text-only}}$*	57.8%$_{\pm 0.0\%}$	43.1%$_{\pm 1.4\%}$	36.4%$_{\pm 3.5\%}$	50.4%$_{\pm 1.4\%}$	63.4%$_{\pm 2.5\%}$	2.3%$_{\pm 0.0\%}$	38.6%$_{\pm 2.3\%}$	24.6%$_{\pm 1.4\%}$
WebVoyager*	51.1%$_{\pm 2.2\%}$	52.9%$_{\pm 1.4\%}$	62.8%$_{\pm 2.3\%}$	52.0%$_{\pm 1.3\%}$	59.3%$_{\pm 3.7\%}$	32.6%$_{\pm 2.7\%}$	47.0%$_{\pm 1.3\%}$	57.9%$_{\pm 2.7\%}$
WebVoyager$_{\text{Claude}}$*	45.9%$_{\pm 3.4\%}$	58.6%$_{\pm 4.2\%}$	58.1%$_{\pm 4.0\%}$	55.0%$_{\pm 7.0\%}$	56.9%$_{\pm 1.4\%}$	19.0%$_{\pm 1.3\%}$	46.2%$_{\pm 1.3\%}$	68.2%$_{\pm 1.3\%}$
WebVoyager$_{\text{GPT-4o}}$*	56.3%$_{\pm 1.3\%}$	53.7%$_{\pm 2.5\%}$	56.6%$_{\pm 1.3\%}$	60.5%$_{\pm 0.0\%}$	57.7%$_{\pm 3.7\%}$	43.9%$_{\pm 3.5\%}$	44.0%$_{\pm 2.7\%}$	65.1%$_{\pm 2.8\%}$

WebVoyager

Findings: The GPT-4V judge agrees with humans 85.3% of the time (κ = 0.70), making live-website evaluation scalable.

	Success Rate	Consistency	
		Agreement	κ
k=1	47.7%	75.3%	0.51
k=2	55.3%	79.7%	0.59
k=3	54.3%	81.3%	0.62
Full	58.3%	85.3%	0.70

Table 2: Consistency between GPT-4V and Human. Success Rate is the overall success rate of all tasks given by GPT-4V. Based on the annotations given by GPT-4V and Human (after alignment), we compute Agreement, i.e., the label overlap, and the Kappa values.

WebVoyager Backbone	Evaluator		
	GPT-4V	Claude-3-Opus	GPT-4o
GPT-4V	57.1	55.1	63.0
Claude-3-Opus	52.8	61.6	55.4
GPT-4o	55.5	54.9	64.1

Table 3: Overall Task Success Rate of WebVoyager using automatic evaluation. We employ GPT-4V, Claude-3-Opus, and GPT-4o as backbones for WebVoyager and run all tasks, followed by automatic evaluation using these three models.

Autonomous Evaluation of Digital Agents

Motivation: Web and device agents only get graded by hand-written benchmark checks. Can a model judge a trajectory, and use that to improve the agent?

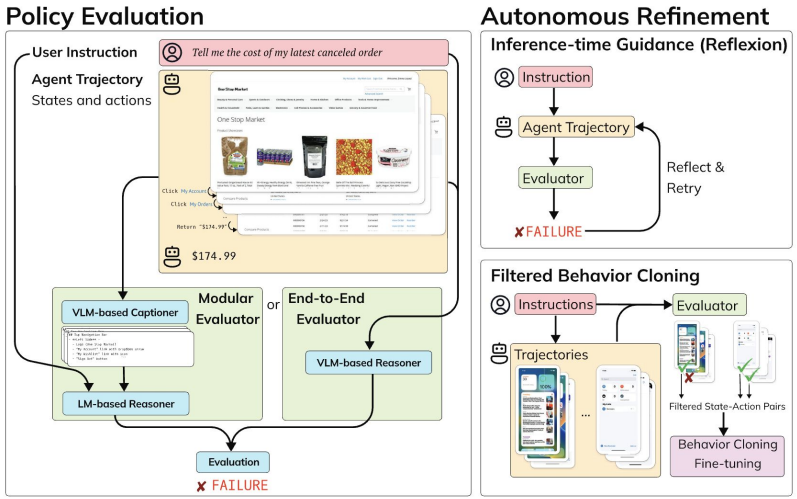


Figure 1: Method overview: A model-based evaluator provides evaluation of a digital agent's trajectory (left). It can be used as the reward function for Reflexion (Shinn et al., 2023) or filtered behavior cloning to enhance model performance (right).

We propose to automatically evaluate user instructions and arbitrary agent trajectories with domain-general neural models. We explore two main variants of this approach (Figure 1, left): first, a modular caption-then-reason approach where a vision-language model (VLM) first captions the screenshots, and a language model (LM) is used to reason about if an agent succeeds based on textual information; and second, an end-to-end approach where we prompt an advanced VLM like GPT-4V (Achiam et al., 2023) to directly evaluate a trajectory. These two different approaches offer trade-offs in performance, cost, and transparency.

Pan et al., "Autonomous Evaluation and Refinement of Digital Agents," COLM 2024

Autonomous Evaluation of Digital Agents

Method: Vision-language evaluators read the instruction plus screenshots of the agent's trajectory and output success or failure.

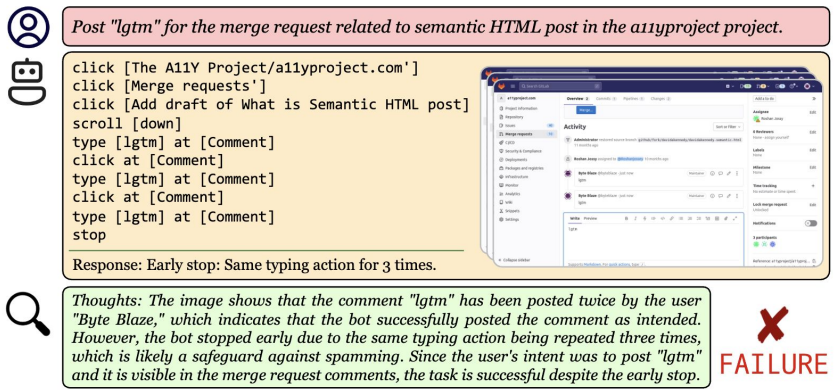


Figure 6: The evaluator correctly judge a task execution as successful despite the task ends with an exception.

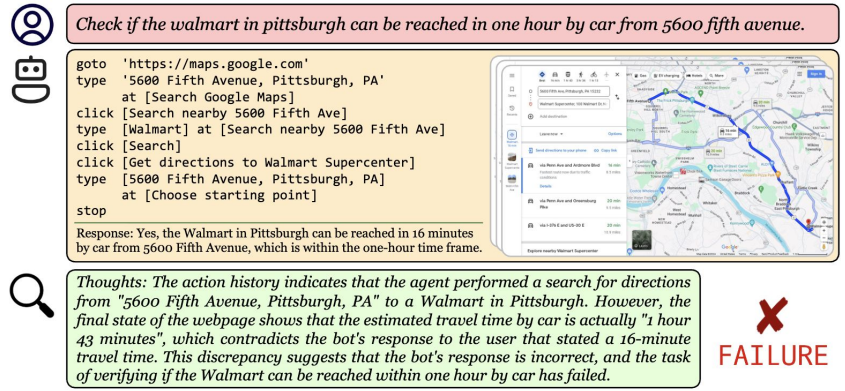


Figure 8: Captioning error where driving time is not captured, and the public transportation time is incorrectly captioned as driving time. The error is corrected if GPT4V is used.

Autonomous Evaluation

Method: Vision-language evaluators read the instruction plus screenshots of the agent's trajectory and output success or failure.

System Prompt
You are an expert in evaluating the performance of a web navigation agent. The agent is designed to help a human user navigate a website to complete a task. Given the user's intent, the agent's action history, the final state of the webpage, and the agent's response to the user, your goal is to decide whether the agent's execution is successful or not.
There are three types of tasks: 1. Information seeking: The user wants to obtain certain information from the webpage, such as the information of a product, reviews, map info, comparison of map routes, etc. The bot's response must contain the information the user wants, or explicitly state that the information is not available. Otherwise, e.g. the bot encounters an exception and respond with the error content, the task is considered a failure. Besides, be careful about the sufficiency of the agent's actions. For example, when asked to list the top-searched items in a shop, the agent should order the items by the number of searches, and then return the top items. If the ordering action is missing, the task is likely to fail. 2. Site navigation: The user wants to navigate to a specific page. Carefully examine the bot's action history and the final state of the webpage to determine whether the bot successfully completes the task. No need to consider the bot's response. 3. Content modification: The user wants to modify the content of a webpage or configuration. Carefully examine the bot's action history and the final state of the webpage to determine whether the bot successfully completes the task. No need to consider the bot's response.
IMPORTANT Format your response into two lines as shown below:
Thoughts: <your thoughts and reasoning process> Status: "success" or "failure"

User Prompt
User Intent: intent
Action History: {last actions}
The last snapshot of the web page is shown in the image.
Bot response to the user: {response if response else "N/A"}.

Figure 12: Prompts to query the VLM Evaluator in WebArena.

System Prompt
You are a GUI Trajectory Evaluator. Your task is to observe a bot's action within a graphical user interface (GUI) and classify its behavior into one of four categories based on its progress towards a specified goal. The categories are:

1. "towards-the-goal" - The bot is moving closer to achieving the goal.
2. "not-sure" - It's unclear if the bot's actions are helping reach the goal.
3. "goal-reached" - The bot has successfully completed the goal.
4. "away-from-the-goal" - The bot's actions are diverting it from the goal.

Please format your response as follows:

Thoughts: [Explain your reasoning here]
Response: "towards-the-goal", "not-sure", "goal-reached", or "away-from-the-goal"

Here are some example responses:

Example 1:
Thoughts: The goal is to 'set an alarm at 8:00 am.' Initially, the bot is on the home screen. After a tap action, it navigates to the alarm app, indicating progress towards the goal.
Response: "towards-the-goal"

Example 2:
Thoughts: The goal is to 'buy the latest iPhone on Amazon.' The bot starts at the checkout page on Amazon. After a tap action, the screen shows a successful purchase, signifying that the goal has been reached.
Response: "goal-reached"

Example 3:
Thoughts: The goal is to 'show me the weather in New York.' The bot begins on London's weather page. After pressing 'home', it returns to the home screen, moving away from the goal.
Response: "away-from-the-goal"

Example 4:
Thoughts: The goal is to 'buy some coffee on the Starbucks app.' The bot begins on the Amazon app. After pressing 'back,' it moves to the home screen, which is a prerequisite for opening the Starbucks app.
Response: "towards-the-goal"

Example 5:
Thoughts: The goal is to 'open YouTube.' The bot begins on the home screen. After a swipe, it appears to remain on the same page, suggesting no progress towards the goal.
Response: "not-sure"

Note:
You should be extra-careful when assigning "goal-reached" or "towards-the-goal" labels. If you are unsure, please select "not-sure" instead.

User Prompt
Goal: {intent}
Original State:
```md
{current_state} ```

State after action: "{action}":
```md
{next_state}
```

Figure 19: Prompts to query the LM Reasoner in iOS for per-step evaluation.
```

Autonomous Evaluation of Digital Agents

Findings: 74.4 to 92.9% agreement with oracle metrics. Used as a reward, they improve a WebArena agent by 29% (relative) and an iOS agent by ~75%.

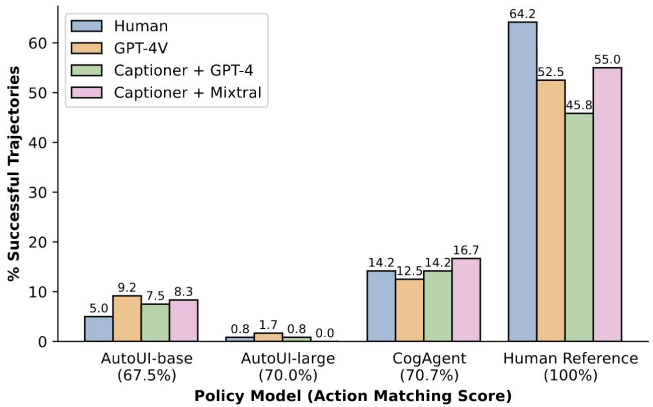


Figure 2: Evaluating models in Android-in-the-Wild with different evaluation methods. We use human judgments of trajectory success as oracle reference and compare it with judgments from our evaluation models and AitW's standard action matching score.

	GPT-4V	QWen-VL-chat	Captioner + Mixtral	Captioner + GPT-4
WebArena (%)	80.6	68.0	74.4	82.1
Android (%)	90.6	70.2	92.9	89.8

Table 1: Comparison of evaluators accuracy against oracle evaluator or human judge in WebArena and Android.

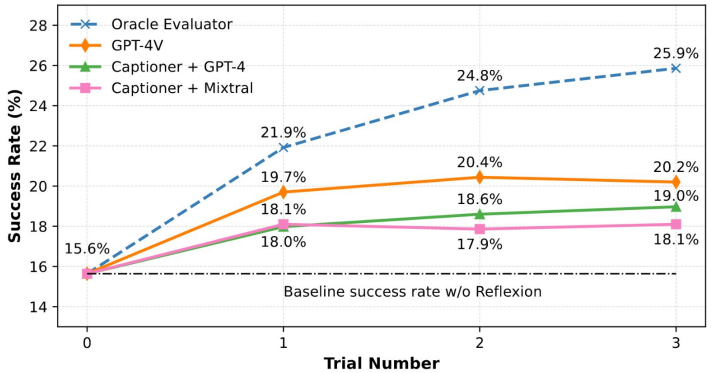


Figure 3: Results of applying Reflexion for up to 3 rounds using different evaluation systems on the WebArena benchmarks. Here, the oracle evaluator denotes performance using WebArena's built-in evaluation functions as the reward function; this provides an upper-bound of improvement using Reflexion.

Agent-as-a-Judge

Motivation: Agents produce workspaces, not just answers. Can judge be an agent?

Contemporary evaluation techniques are inadequate for agentic systems. These approaches either focus exclusively on final outcomes—ignoring the step-by-step nature of agentic systems, or require excessive manual labour. To address this, we introduce the Agent-as-a-Judge framework, wherein agentic systems are used to evaluate agentic systems. This is an organic extension of the LLM-as-a-Judge framework, incorporating agentic features that enable intermediate feedback for the entire task-solving process. We apply the Agent-as-a-Judge to the task of code generation. To overcome issues with existing benchmarks and provide a proof-of-concept testbed for Agent-as-a-Judge, we present DevAI, a new benchmark of 55 realistic automated AI development tasks. It includes rich manual annotations, like a total of 365 hierarchical user requirements. We benchmark three of the popular agentic systems using Agent-as-a-Judge and find it dramatically outperforms LLM-as-a-Judge and is as reliable as our human evaluation baseline.

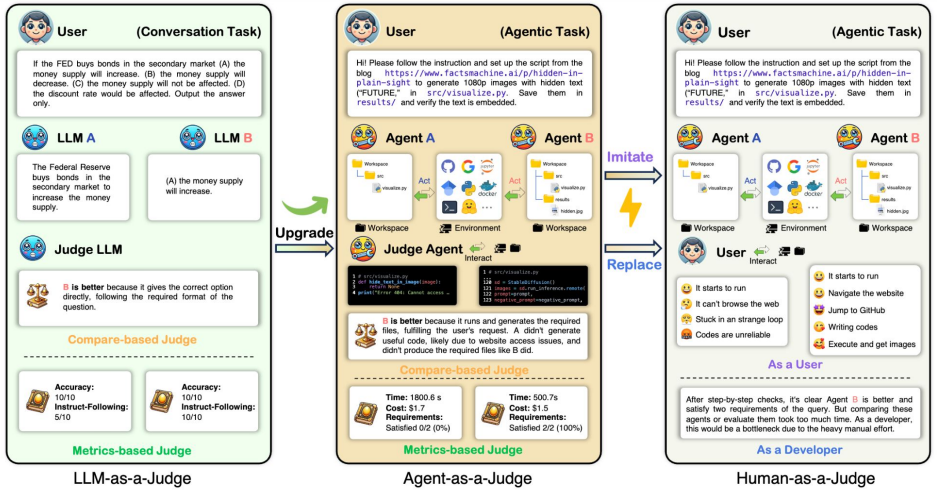
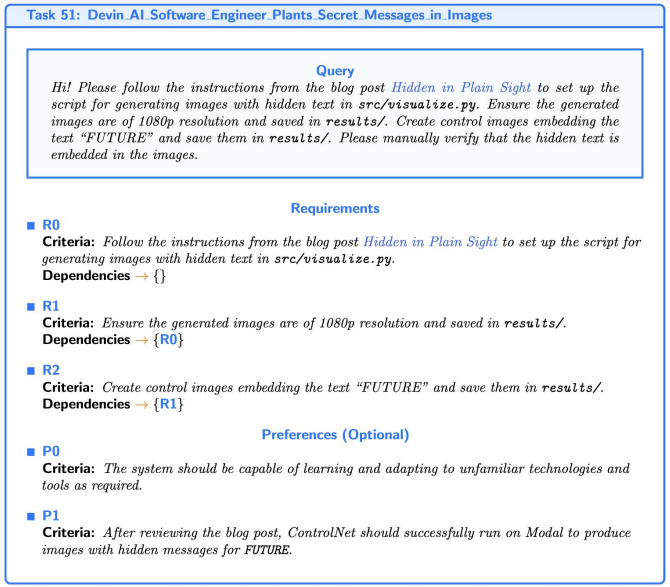


Figure 1 We introduce the Agent-as-a-Judge framework wherein agentic systems are used to evaluate agentic systems. We compare this to LLM-as-a-Judge, which uses LLMs to evaluate LLMs and for which Agent-as-a-Judge is a natural evolution, and Human-as-a-Judge, where skilled human labourers manually evaluate an agentic system.

Zhuge et al., "Agent-as-a-Judge: Evaluate Agents with Agents," ICML 2025

Agent-as-a-Judge

Method: DevAI: 55 AI-dev tasks with 365 hierarchical requirements. Judge agent explores workspace (locate, read, search, check the trajectory) to verify.



Task 51: Devin_AI_Software_Engineer_Plants_Secret_Messages_in_Images

Query

Hi! Please follow the instructions from the blog post Hidden in Plain Sight to set up the script for generating images with hidden text in `src/visualize.py`. Ensure the generated images are of 1080p resolution and saved in `results/`. Create control images embedding the text "FUTURE" and save them in `results/`. Please manually verify that the hidden text is embedded in the images.

Requirements

- **R0**
  Criteria: *Follow the instructions from the blog post Hidden in Plain Sight to set up the script for generating images with hidden text in `src/visualize.py`.*
  Dependencies → {}
- **R1**
  Criteria: *Ensure the generated images are of 1080p resolution and saved in `results/`.*
  Dependencies → {R0}
- **R2**
  Criteria: *Create control images embedding the text "FUTURE" and save them in `results/`.*
  Dependencies → {R1}

Preferences (Optional)

- **P0**
  Criteria: *The system should be capable of learning and adapting to unfamiliar technologies and tools as required.*
- **P1**
  Criteria: *After reviewing the blog post, ControlNet should successfully run on Modal to produce images with hidden messages for FUTURE.*

Motivated by the ideas outlined above, we propose the DevAI dataset. DevAI consists of a curated set of 55 tasks, each defined by (1) a plain text user query that describes an AI development task; (2) a set of plain text requirements (for a total of 365 requirements), each with a set of dependencies connecting them to other requirements; and (3) a set of preferences (for a total of 125 preferences) which represent softer requirements.

Table 2 Human-as-a-Judge for AI Developers. (I) and (D) represent independent performance versus performance considering task dependencies. 👤 indicates multiple experts evolved, and ⬜ means the evaluations use white-box testing (allowing access to the generated workspace, human-collected trajectories, and open-source codebases). The results were derived from expert judgments and deliberations (see Appendix H).

Metric	MetaGPT (Hong et al., 2024b)	GPT-Pilot (Pythagora.io, 2023)	OpenHands (Wang et al., 2024d)
	👤 / ⬜ Human-as-a-Judge		
(A) Requirements Met (I)	22.13%	44.80%	42.89%
(B) Requirements Met (D)	6.55%	28.96%	28.68%
(C) Self-Termination	41.81%	5.45%	54.54%
(D) Task Solve Rate	0.00%	1.81%	1.81%

Agent-as-a-Judge

Findings: ~90% agreement with human experts vs. ~70% for LLM-as-a-Judge, at ~97% less time and cost than a panel of three experts.

Metric	MetaGPT (Hong et al., 2024b)	GPT-Pilot (Pythagora.io, 2023)	OpenHands (Wang et al., 2024d)
LLM-as-a-Judge			
(a) Requirements Met (I)	19.39% (2.74%)	12.56% (32.24%)	11.47% (31.42%)
(b) Requirements Met (D)	1.63% (4.92%)	4.09% (24.87%)	2.18% (26.50%)
(c) Task Solve Rate	0.0% (0.0%)	0.0% (1.81%)	0.0% (1.81%)
Alignment Rate ↑	84.15%	65.30%	60.38%
Agent-as-a-Judge			
(I) Requirements Met (I)	25.40% (3.26%)	53.00% (8.20%)	42.62% (0.27%)
(II) Requirements Met (D)	5.73% (0.81%)	39.89% (10.93%)	26.50% (2.17%)
(III) Task Solve Rate	0.0% (0.0%)	5.45% (3.64%)	1.81% (0.00%)
Alignment Rate ↑	88.52%	83.88%	90.44%

[Phase 3] Human Actions

Phase 3: **can an LM judge an agent's actions and their effects, not just its words?**

- Can an LM tell whether an agent's trajectory succeeded?

  WebVoyager: VLMs reading screenshots agree with humans 85% of the time, and with oracle checks 74–93%.

- Can the judge act like an agent, inspecting the environment itself?

  Agent-as-a-Judge: letting the judge explore the workspace raises agreement with experts from ~70% to ~90%.

[Phase 3] Human Actions

Phase 3: **can an LM judge an agent's actions and their effects, not just its words?**

- How reliable are these judges compared with human experts?

Good enough to train with (Pan: +29% on WebArena), but off by 10+ points on some slices (WebVoyager's per-site gaps)

Biases + Validating Judgment

Biases + Validating Judgment

LM judges may have arbitrary biases. Let's take a look at some of them + how we've dealt with them.

- LM judges may be biased by the **order** of answers.
- LM judges may be biased by the **length** of answers.
- How do we build a judge that matches human judgment for my task?

LLMs Are Not Fair Evaluators

Motivation: When a judge compares 2 answers, does it matter which is shown first?

In this paper, we uncover a systematic bias in the evaluation paradigm of adopting large language models (LLMs), e.g., GPT-4, as a referee to score and compare the quality of responses generated by candidate models. We find that the quality ranking of candidate responses can be easily hacked by simply altering their order of appearance in the context. This manipulation allows us to skew the evaluation result, making one model appear considerably superior to the other, e.g., Vicuna-13B could beat ChatGPT on 66 over 80 tested queries with ChatGPT as an evaluator.

[Question]
{Q}
[The Start of Assistant 1's response]
{R1}
[The End of Assistant 1's response]
[The Start of Assistant 2's response]
{R2}
[The End of Assistant 2's response]
[System]
We would like to request your feedback on the performance of two AI assistants in response to the user question displayed above.
Please rate the helpfulness, relevance, accuracy, level of details of their responses. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance.
Please first output a single line containing only two values indicating the scores for Assistant 1 and 2, respectively.
The two scores are separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment.

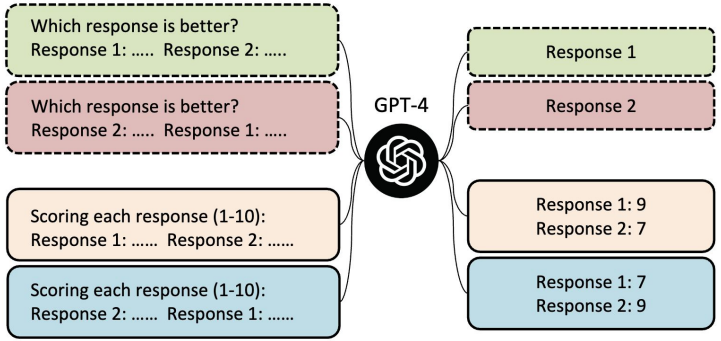


Figure 1: Simply changing the order of candidate responses leads to overturned comparison results, even though we add the command "ensuring that the order in which the responses were presented does not affect your judgment" into the prompt.

Wang et al., "Large Language Models are not Fair Evaluators," ACL 2024 (arXiv May 2023)

LLMs Are Not Fair Evaluators

Method: DevAI: 55 AI-dev tasks with 365 hierarchical requirements. Judge agent explores workspace (locate, read, search, check the trajectory) to verify.

To address this issue, we propose a calibration framework with three simple yet effective strategies: 1) Multiple Evidence Calibration, which requires the evaluator model to generate multiple evaluation evidence before assigning ratings; 2) Balanced Position Calibration, which aggregates results across various orders to determine the final score; 3) Humanin-the-Loop Calibration, which introduces a balanced position diversity entropy to measure the difficulty of each example and seeks human assistance when needed.

Evaluators	Vicuna-13B v.s. Other Models	Vicuna-13B Win Rate		Conflict Rate
		as Assistant1	as Assistant2	
GPT-4	Vicuna-13B v.s. ChatGPT	51.3%	23.8%	37 / 80 (46.3%)
GPT-4	Vicuna-13B v.s. Alpaca-13B	92.5%	92.5%	4 / 80 (5.0%)
ChatGPT	Vicuna-13B v.s. ChatGPT	2.5%	82.5%	66 / 80 (82.5%)
ChatGPT	Vicuna-13B v.s. Alpaca-13B	37.5%	90%	42 / 80 (52.5%)

Table 2: The Win Rate of Vicuna-13B significantly fluctuates when positioned as Assistant 1 and Assistant 2, with GPT-4 and ChatGPT serving as evaluators. Conflict Rate refers to the proportion of conflicting results given by the same evaluator when simply changing the position of two models.

$$\textbf{Conflict Rate} = \frac{\sum_{i=1}^{N} \mathbb{I}(\mathbf{ER}_i^{r12} \neq \mathbf{ER}_i^{r21})}{N}, \quad (1)$$

LLMs Are Not Fair Evaluators

Method: DevAI: 55 AI-dev tasks with 365 hierarchical requirements. Judge agent explores workspace (locate, read, search, check the trajectory) to verify.

To address this issue, we propose a calibration framework with three simple yet effective strategies: 1) Multiple Evidence Calibration, which requires the evaluator model to generate multiple evaluation evidence before assigning ratings; 2) Balanced Position Calibration, which aggregates results across various orders to determine the final score; 3) Humanin-the-Loop Calibration, which introduces a balanced position diversity entropy to measure the difficulty of each example and seeks human assistance when needed.

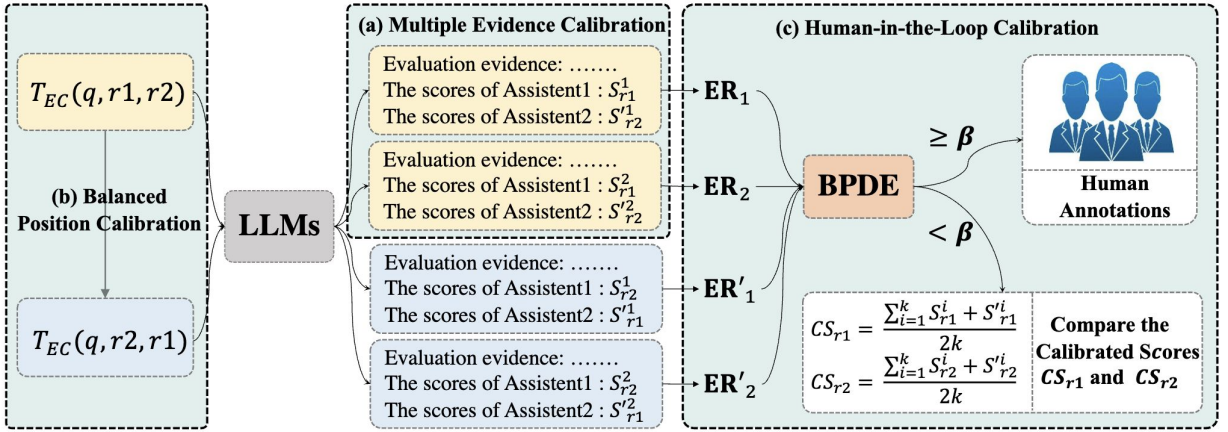


Figure 3: Demonstration of our calibration framework with three calibration methods. S_r and S'_r denotes scores of the response r in the first and second positions, respectively. BPDE is short for Balanced Position Diversity Entropy score, which is calculated based on the evaluation results (ER) of MEC and BPC.

LLMs Are Not Fair Evaluators

Findings: ~90% agreement with human experts vs. ~70% for LLM-as-a-Judge, at ~97% less time and cost than a panel of three experts.

Evaluators	Methods	Accuracy	Kappa	Cost
Human 1	-	68.8%	0.50	$30.0
Human 2	-	76.3%	0.62	$30.0
Human 3	-	70.0%	0.50	$30.0
Human Average	-	71.7%	0.54	$30.0
GPT-4	Vanilla	52.7%	0.24	$2.00
GPT-4	EC ($k = 1$)	56.5%	0.29	$2.00
GPT-4	MEC ($k = 3$)	58.7%	0.30	$3.19
GPT-4	MEC ($k = 6$)	60.9%	0.33	$6.38
GPT-4	MEC ($k = 3$) + BPC ($k = 3$)	62.5%	0.37	$6.38
GPT-4	MEC ($k = 3$) + BPC ($k = 3$) + HITLC ($\beta = 20\%$)	73.8%	0.56	$23.1
ChatGPT	Vanilla	44.4%	0.06	$0.10
ChatGPT	EC ($k = 1$)	52.6%	0.23	$0.10
ChatGPT	MEC ($k = 3$)	53.2%	0.24	$0.17
ChatGPT	MEC ($k = 6$)	55.6%	0.27	$0.34
ChatGPT	MEC ($k = 3$) + BPC ($k = 3$)	58.7%	0.31	$0.34
ChatGPT	MEC ($k = 3$) + BPC ($k = 3$) + HITLC ($\beta = 20\%$)	71.3%	0.52	$18.3

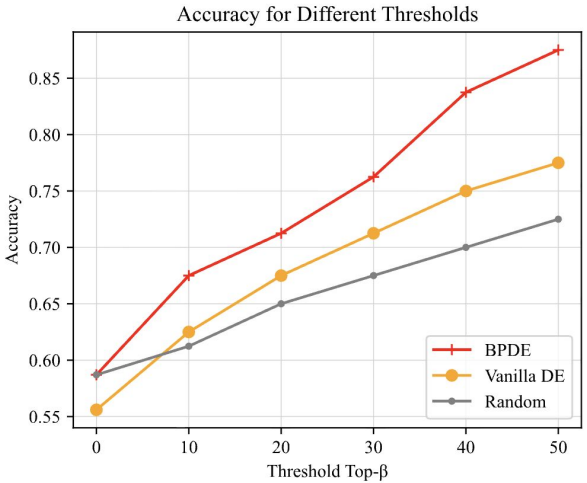
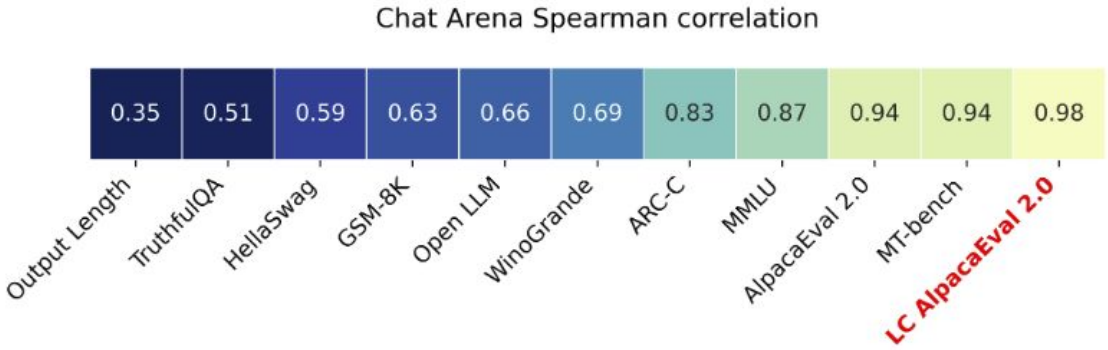


Figure 5: The accuracy of various methods changes with different human assistant thresholds (Top-β) when ChatGPT is used as the evaluator.

Length-Controlled AlpacaEval

Motivation: LLM judges favor longer answers, and models learned to exploit it. Can we remove a known bias after the fact?

> Despite being highly correlated with human preferences, AlpacaEval is known to favor models that generate longer outputs. We introduce a length-controlled AlpacaEval that aims to answer the counterfactual question: "What would the preference be if the model's and baseline's output had the same length?"



Dubois et al., "Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators," COLM 2024

Length-Controlled AlpacaEval

Method: Fit regression that predicts judge's preference from LM, the instruction, and the length difference; then ask what it would prefer if lengths were equal.

Motivated by the previous discussion, we will model the AlpacaEval predictions $f(z_m, z_b, x)$ with a logistic regression that has 3 terms: model, length, and instruction.

$$q_{\theta,\phi,\psi}(y = 1 | z_m, z_b, m, b, x) := \text{logistic}\left(\underbrace{\theta_m - \theta_b}_{\text{Model}} + \underbrace{\phi_{m,b} \cdot \tanh\left(\frac{\text{len}(z_m) - \text{len}(z_b)}{\text{std}(\text{len}(z_m) - \text{len}(z_b))} \right)}_{\text{Length}} + \underbrace{(\psi_m - \psi_b)\gamma_x}_{\text{Instruction}} \right).$$

Length-Controlled AlpacaEval

Findings: Much harder to game with verbosity, and Spearman correlation with Chatbot Arena rises from 0.94 to 0.98.

	AlpacaEval			Length-controlled AlpacaEval		
	concise	standard	verbose	concise	standard	verbose
gpt4_1106_preview	22.9	50.0	64.3	41.9	50.0	51.6
Mixtral-8x7B-Instruct-v0.1	13.7	18.3	24.6	23.0	23.7	23.2
gpt4_0613	9.4	15.8	23.2	21.6	30.2	33.8
claude-2.1	9.2	15.7	24.4	18.2	25.3	30.3
gpt-3.5-turbo-1106	7.4	9.2	12.8	15.8	19.3	22.0
alpaca-7b	2.0	2.6	2.9	4.5	5.9	6.8

Figure 3: Length-controlled AlpacaEval decreases the sensitivity to prompting the evaluated model for more concise or verbose outputs.

Length-Controlled AlpacaEval

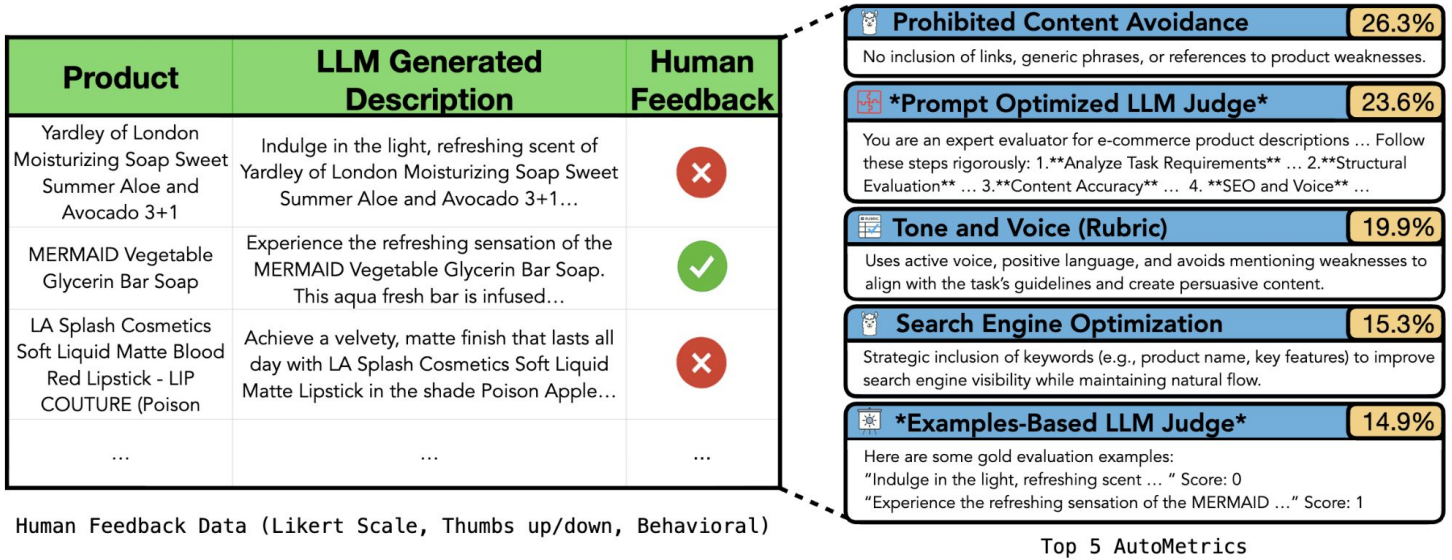
Findings: Much harder to game with verbosity, and Spearman correlation with Chatbot Arena rises from 0.94 to 0.98.

Table 1: Length-controlled win rate has the best Chatbot Arena correlation and gameability from considered methods, while still being relatively robust to adversarial attacks. Each row is a different metric we considered.

	Chatbot Arena correlation (↑)	Gameability (↓)	Adversarial win rate gain (↓)
Win rate	0.94	26%	0.0
Length-controlled	0.98	10%	8.5
Length-normalized	0.96	15%	3.6
Length-balanced	0.95	15%	40.8

AutoMetrics

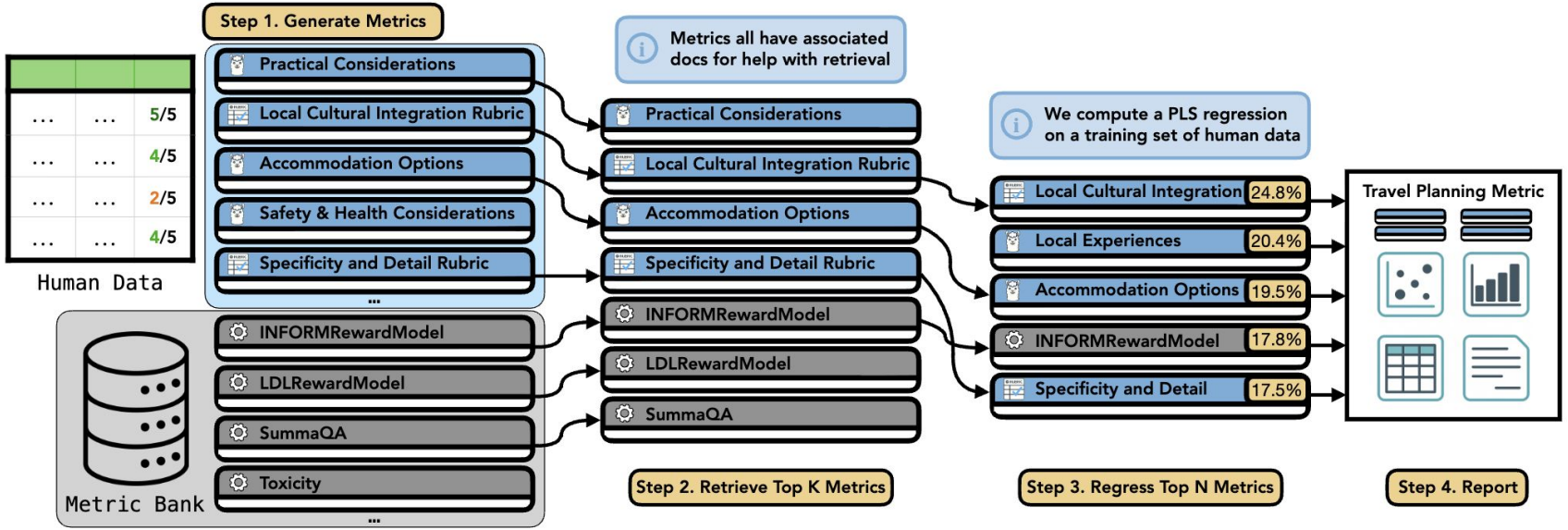
Motivation: Real user feedback is the gold standard, but it's scarce for new tasks. Can a handful of human labels produce a judge that matches their judgment?



Ryan et al., "AutoMetrics: Approximate Human Judgments with Automatically Generated Evaluators," ICLR 2026

AutoMetrics

Method: Get candidates from 48 existing metrics (MetricBank) plus auto-generated LLM-judge criteria. Fit a regression to the human labels to pick and weight them.



AutoMetrics

Findings: +33.4% Kendall correlation with human ratings than LLM-as-a-Judge, from fewer than 100 labels; also works as a proxy reward as well as a verifiable one.

In order to evaluate our AutoMetrics method, we collect two types of tasks: In-Distribution Tasks, which are tasks where some of the metrics in our Metric Bank were designed to directly evaluate the task, and Out-of-Distribution Tasks, which are tasks where no metric in particular was designed to assess the task. All of our tasks utilize human feedback for evaluation, encompassing behavioral feedback, binary feedback (thumbs up/down), and Likert scale feedback, which is already collected as part of the dataset.

Method	In-Distribution		Out-of-Distribution		
	SimpEval	HelpSteer2	EvalGen	RealHumanEval	CoGym
Model Agnostic					
Best Existing Metric	0.246 ± 0.00	0.327 ± 0.00	0.193 ± 0.00	0.138 ± 0.00	0.074 ± 0.00
MetaMetrics (Winata et al., 2025)	0.127 ± 0.01	0.204 ± 0.00	-0.214 ± 0.01	0.025 ± 0.01	-0.119 ± 0.02
Finetuned LLM	0.076 ± 0.08	0.039 ± 0.03	0.054 ± 0.05	0.049 ± 0.06	0.223 ± 0.20
GPT-4o-mini Backbone					
LLM-Judge	0.272 ± 0.02	0.259 ± 0.01	0.161 ± 0.14	0.069 ± 0.01	**0.199** ± 0.13
DnA Eval (Li et al., 2025)	0.234 ± 0.03	0.255 ± 0.02	0.174 ± 0.16	0.152 ± 0.01	0.185 ± 0.10
AutoMetrics (Ours)	**0.321** ± 0.04	**0.324** ± 0.01	**0.334** ± 0.06	**0.160** ± 0.00	-0.034 ± 0.17
Qwen-3-32B Backbone					
LLM-Judge	0.294 ± 0.04	0.334 ± 0.02	0.272 ± 0.13	0.025 ± 0.01	0.276 ± 0.19
DnA Eval (Li et al., 2025)	0.042 ± 0.04	0.260 ± 0.02	0.232 ± 0.19	0.071 ± 0.15	0.353 ± 0.25
AutoMetrics (Ours)	**0.316** ± 0.02	**0.342** ± 0.01	**0.382** ± 0.05	**0.145** ± 0.00	**0.365** ± 0.08

Biases + Validating Judgment

- LM judges may be biased by the **order** of answers.

  Position, length, self-preference: errors that always lean the same way, so more examples don't cancel them out.

- LM judges may be biased by the **length** of answers.

  Wang et al. 2023 (position), Length-Controlled AlpacaEval (length)

- How do we build a judge that matches human judgment for my task?

  AutoMetrics: build and check a judge against a small set of human labels.

Handling Nondeterminism

Handling Nondeterminism

Recall: Let's say you run a task 5 times, and get: ❌ ✅ ❌ ❌ ✅

How do you report this outcome?

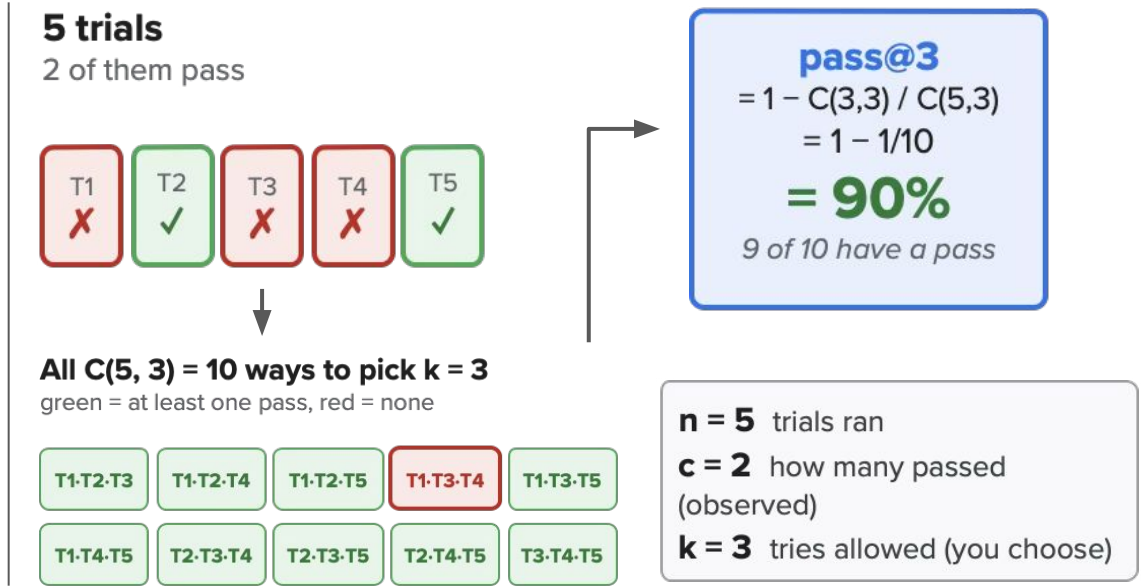
- **Average (40%)**: Communicates expected success rate per try.
  ⇒ *Hides*: So is each attempt a coin flip? Or tasks are always all pass/fail?
- **At least once (Yes)**: Says whether agent is capable of solving task *at all*.
  ⇒ *Hides*: Luck + cost of retrying. What if you scale arbitrarily to k retries?
- **Solved every time (No)**: Says whether you can count on agent
  ⇒ *Hides*: Near-misses and potential infra failings. Goes towards 0% w/ more k.

pass@k

Introduced in HumanEval (Chen et al 2023), pass@k = “probability that at least one of k independent solution attempts will succeed”.

> [W]e generate n ≥ k samples per task (in this paper, we use n = 200 and k ≤ 100), count the number of correct samples c ≤ n which pass unit tests, and calculate the unbiased estimator

$$\text{pass@}k := \mathop{\mathbb{E}}_{\text{Problems}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right]$$

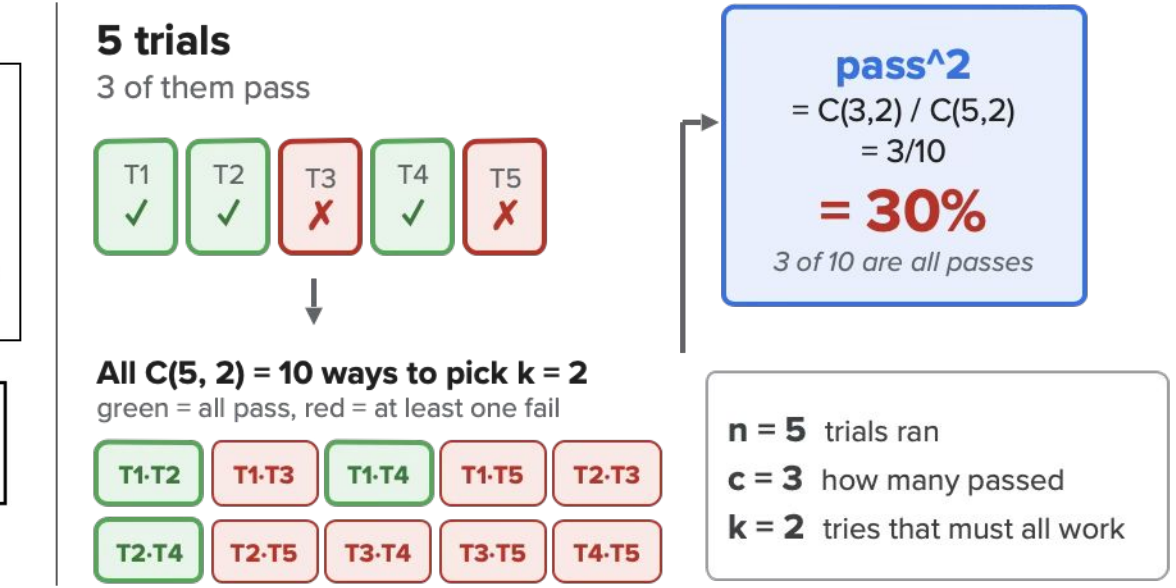


pass^k

Introduced in tau-bench (Yao et al 2024), pass@k = “of all the ways to pick k of n trials, what fraction of these combinations are all passes?”.

For real-world agent tasks requiring reliability and consistency like customer service, we propose a new metric – pass^k (pass hat k), defined as the chance that all ki.i.d. task trials are successful, averaged across tasks. Therefore, if a task is run for n trials and c of those trials end up successful (r = 1), unbiased estimates for pass^k and pass@k would be:

$$\text{pass^k} = \mathbb{E}_{\text{task}} \left[\binom{c}{k} \Big/ \binom{n}{k} \right]$$



Why not just report the average?

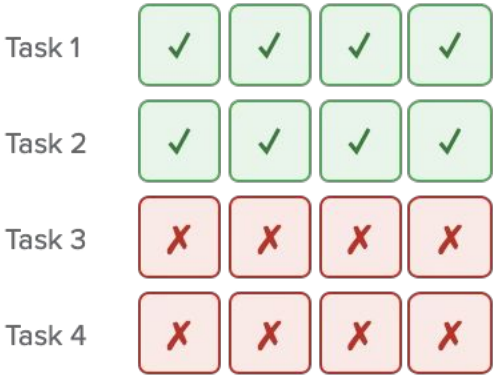
Two agents can share the same average and behave completely differently.

Agent A
a coin flip on every task



4 trials per task

Agent B
2 tasks always, 2 tasks never



4 trials per task

	Average	pass@2	pass^2
A	50%	**83%**	**17%**
B	50%	50%	50%

A: retrying helps a lot, but you can't count on any single run.

B: retrying doesn't help, but what it solves, it solves every time.

Which one should you use?

It really depends on how your agent is being used.

pass@k

You can retry, and a check picks the winner.

- Coding with tests: sample k patches, keep one that passes
- Research or Search: any correct find is enough

pass^k

Every attempt reaches a user or changes the world.

- Customer service: each customer gets 1 try
- Generally, if agents take real actions (purchases, emails)

Either way: report k + number of trials n. Without k, the number is meaningless.