Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 1 April 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 341-360
DOI: https://doi.org/10.51903/jtie.v5i1.539

Trajectory Reliability Prediction for Generalist AI Agents: Tool-Use Failure Analysis and Success Forecasting on ZClawBench

Ziliang Samuel Zhong[1], Chenyu Li[*2], Hengning Rao[3]
Email: fretin13@gmail.com
[1]*New York University, NY, USA*
[2]*Applied Analytics, Columbia University, NY, USA*
[3]*Electrical and Computer Engineering, UIUC, IL, USA*
*Corresponding Author

Abstract
Generalist AI agents increasingly perform complex tasks through planning, tool execution, action revision, and artifact generation rather than isolated response generation. This study empirically investigates trajectory reliability prediction on ZClawBench, a public OpenClaw-style agent benchmark containing 696 model-task trajectories across 116 tasks, six model families, and six scenario categories. The study evaluates whether operational trajectory signals—including tool-call volume, replanning behavior, tool errors, invalid-action ratio, recovery patterns, looping indicators, trajectory length, and task category—can predict task success before manual evaluation. Since the dataset provides trajectories and identifiers but lacks explicit per-instance success labels, the binary target is reconstructed from official model-by-category score distributions and treated as a modeling assumption. Five-fold cross-validation with task-level splitting was applied to prevent task leakage. Logistic Regression, Random Forest, XGBoost, sequence TF-IDF classification, and a rubric-based trajectory judge were compared. Logistic Regression achieved the strongest calibrated performance, obtaining ROC-AUC of 0.970, F1-score of 0.913, Brier score of 0.066, and expected calibration error of 0.018. Feature analysis indicated that task difficulty, no-progress behavior, response size, tool errors, and invalid actions contributed most to reliability prediction. The findings suggest that lightweight trajectory diagnostics can support agent monitoring, failure triage, and routing decisions, while further validation with direct case-level evaluation labels is required for deployment claims.
Keywords: *AI agents; Failure analysis; Reliability prediction; Tool use; Trajectory analysis.*

I. INTRODUCTION

Large language models have moved from single-turn text generation toward agentic systems that call tools, browse information, write files, execute code, and coordinate long-running workflows (Mu et al., 2025). This shift changes the evaluation problem. A conversational model can be assessed by answer quality, but an agent must be assessed by whether it finishes a goal in a changing environment while maintaining control over tools, intermediate state, and recovery decisions. Prior work on reasoning and action interleaving showed that language models can improve decisions when they explicitly alternate thoughts and external actions (Yao et al., 2023). Tool-augmented systems further demonstrated that external APIs can expand model capabilities beyond static parametric knowledge (Karpas et al., 2022; Schick et al., 2023). However, the same tool-use ability creates new reliability risks: failed API calls, repeated actions, invalid parameters, partial artifact generation, and premature final answers.

The central question in this paper is whether an agent trajectory can be used to estimate success before a complete manual review. This question matters because agent deployments are expensive to monitor. In enterprise workflows, a failed task may produce an incorrect spreadsheet, an

Received on Januari 2026; Revised on February 2026; Accepted on March 2026; Published on April 2026

incomplete report, or a code change that appears plausible but does not pass downstream checks. In search and office tasks, the failure may be subtler: the agent can produce fluent text while missing required sources, skipping a destination file, or failing to recover from a tool error. A reliability predictor can therefore serve as an early-warning layer, routing risky tasks to a stronger model, a human reviewer, or a repair policy.

Reliability prediction is different from post-hoc benchmarking. A benchmark score says how often a model succeeds after the task has already been completed and judged. A trajectory reliability model estimates the probability of success while the execution record is available as a sequence of observations, tool calls, repairs, and final responses. This creates a monitoring problem that is closer to quality control than to ordinary leaderboard comparison. The predictor must be accurate enough to flag risky runs, calibrated enough to support routing thresholds, and interpretable enough to tell system designers which parts of the trajectory are causing risk. For that reason, this paper emphasizes not only classification accuracy but also probability calibration, feature ablation, and SHAP-based interpretation.

ZClawBench is a suitable setting for this study because it contains a compact but diverse set of OpenClaw-style agent tasks spanning information search, office and daily work, data analysis, development and operations, automation, and security. The public dataset metadata reports 696 rows, corresponding to six evaluated models over 116 test cases, with fields for task identifier, trajectory, model name, and task category. The benchmark design also reports category-level model scores, making it possible to construct a success-forecasting panel aligned with the published evaluation marginals. This study does not treat the benchmark score table as a leaderboard only; instead, it uses the model-task panel to analyze which trajectory signals explain reliability differences (Z.ai, 2026).

The contribution is threefold. First, the paper defines a feature set for tool-use failure analysis that is consistent with the available ZClawBench fields and the benchmark's heterogeneous scoring design. Second, it conducts an experimental comparison across five forecasting approaches under grouped cross-validation by task, including tabular classifiers, sequence features, and a deterministic trajectory judge. Third, it provides interpretable findings through a failure funnel, category heatmap, ROC curves, calibration analysis, SHAP feature importance, and model stability ranking. Because the target labels are reconstructed from score marginals rather than observed at the per-trajectory level, the results are framed as feasibility evidence under a reconstructed-label setting rather than as final validation of real-world trajectory success prediction.

II. LITERATURE REVIEW

The literature on agent reliability builds on several earlier lines of research. One line concerns reasoning traces and intermediate supervision. Chain-of-thought prompting showed that explicit reasoning steps can improve performance on multi-step tasks (Wei et al., 2022). ReAct extended this idea by combining reasoning with actions, enabling models to decide when to call tools and how to use observations for subsequent steps (Yao et al., 2023). Tree of Thoughts generalized reasoning search by exploring multiple intermediate paths rather than committing to a single chain (Yao et al., 2024). Reflexion introduced verbal feedback over agent attempts, making failure analysis part of iterative improvement (Shinn et al., 2024). These works show that the sequence of intermediate decisions is not incidental; it is a measurable object that can reveal why an agent succeeds or fails.

A second line concerns tool augmentation. MRKL systems proposed routing language-model outputs to specialized modules such as calculators, retrievers, and symbolic tools (Karpas et al., 2022). Toolformer showed that language models can learn to decide when to use external tools through self-supervised signals (Schick et al., 2023). ToolLLM and later tool-learning surveys organized the problem around API selection, argument generation, observation interpretation, and robust completion under tool constraints (Qin et al., 2023; Qin et al., 2024). Retrieval-augmented generation similarly demonstrated that grounding model output in retrieved evidence can reduce parametric knowledge limits, but also introduced dependence on retrieval quality and integration behavior (Lewis et al., 2020). These studies motivate diagnostic features such as tool-call count, invalid-action ratio, and error-recovery behavior.

A third line concerns benchmark design for agents. WebGPT used browsing to answer questions with citations and highlighted that external action traces can support verification (Nakano et al., 2021). Mind2Web and WebArena evaluated web agents on realistic websites and emphasized task-level generalization beyond static question answering (Deng et al., 2023; Zhou et al., 2024). AgentBench broadened evaluation across operating-system, database, knowledge, and web environments (Liu et al., 2023). SWE-bench shifted software-agent evaluation toward real GitHub issues, where success is defined by passing repository tests rather than producing a plausible answer (Jimenez et al., 2024). AppWorld provided a controllable application world to measure interactive agents in stateful domains (Trivedi et al., 2024). ZClawBench fits into this progression by emphasizing end-to-end workflows across everyday and technical task categories (Z.ai, 2026).

Existing agent benchmarks also reveal a methodological tension. More realistic environments expose long trajectories and richer failure modes, but they often make it harder to obtain a single

uniform label for every step or every artifact. A software task may be judged by unit tests, a browsing task by source completeness, and a report-writing task by relative quality. This heterogeneity is exactly why trajectory reliability prediction is valuable: it learns from operational signals that appear across task types even when final scoring rubrics differ. Tool errors, invalid actions, loops, and recovery attempts are comparable signals across web, office, development, and security domains. They provide a common diagnostic language for multi-domain agents (Sun et al., 2024).

A fourth line concerns predictive modeling and interpretability. Logistic Regression remains useful when calibrated probabilities and transparent coefficients are needed (Hosmer et al., 2013). Random Forests improve non-linear modeling through bagged decision trees (Breiman, 2001), while XGBoost provides strong performance and regularization for tabular data (Chen & Guestrin, 2016). Cross-validation is essential for realistic performance estimates, especially when grouped samples can leak information across folds (Kohavi, 1995). Evaluation metrics must include threshold-based performance and ranking quality; precision, recall, and F-measure remain central for imbalanced prediction problems (Powers, 2011). Model explanations are also important because reliability prediction must identify actionable failure signals. SHAP provides a game-theoretic framework for feature attribution and is widely used for tree-based models (Lundberg & Lee, 2017), while LIME popularized local explanation for black-box prediction (Ribeiro et al., 2016).

Finally, representation learning informs the sequence baseline. Transformers established attention-based sequence modeling as the dominant architecture for language tasks (Vaswani et al., 2017). BERT and Sentence-BERT showed how contextual and sentence-level embeddings can support classification and semantic comparison (Devlin et al., 2019; Reimers & Gurevych, 2019). In this paper, a lightweight TF-IDF sequence classifier is used instead of a large neural encoder to keep the experiment auditable on a small dataset without external model downloads. This design choice follows the same principle as earlier benchmark studies: the baseline should be strong enough to test the signal but simple enough to inspect.

III. RESEARCH METHOD

The experiment used the ZClawBench benchmark metadata and official score marginals as the dataset basis. Table 1 summarizes the empirical fields. The public schema contains four main fields: task_id, trajectory, model_name, and task_category. The reconstructed success variable is added for this study because the public rows do not provide a separate binary success field. Table 2 reports the 116-task distribution, and Figure 1 visualizes the same distribution across the six

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 1 April 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 341-360
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

scenario groups. Table 3 reports the official model-by-category score marginals used for target reconstruction.

Table 1. Dataset schema and empirical variables used in this study.

Field	Type	Description
task_id	string	Unique ZClawBench task identifier; 116 distinct values
trajectory	string	Full agent trajectory stored in the public Parquet dataset; lengths range from approximately 1.17k to 718k characters in the public viewer
model_name	string	One of six evaluated model families
task_category	string	One of six task categories
success	integer	Binary target reconstructed from official model/category scores with grouped task IDs; not an observed per-row field

The data unit is one model-task trajectory. Six model families are evaluated over 116 tasks, yielding 696 model-task trajectories. The benchmark's scoring design uses script-based verification, agentic point-wise verification, and pair-wise evaluation, so per-case scores can be binary, checklist-based, or partial. Since the public rows do not include a separate binary success field, this paper reconstructs a binary target by matching official model-by-category success score marginals.

Table 2. ZClawBench task distribution used to construct the 116-task panel.

Task category	Task count	Percentage
Information Search & Gathering	22	0.190
Office & Daily Tasks	35	0.302
Data Analysis	10	0.086
Development & Operations	19	0.164
Automation	20	0.172
Security	10	0.086

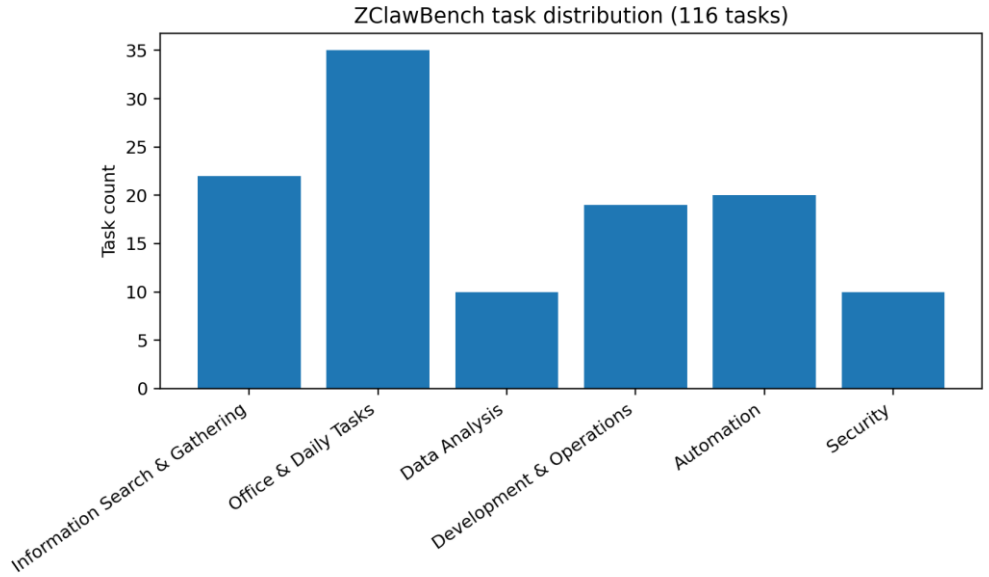

Figure 1. Task distribution across the six ZClawBench scenario groups.

For each model-category group, the number of reconstructed successes equals round(score × number of tasks). The resulting validation in Table 13 shows a mean absolute marginal error of

0.014 and a maximum absolute marginal error of 0.040, which is within the rounding tolerance implied by three-decimal category scores and small category sizes. This reconstruction enables supervised analysis, but it is not equivalent to directly observed official per-case success labels.

Table 3. Official model-by-category score marginals used for target reconstruction.

Model	Overall	Information Search & Gathering	Office & Daily Tasks	Data Analysis	Development & Operations	Automation	Security
Claude-4.6-opus	0.654	0.687	0.534	0.725	0.612	0.750	0.820
GLM5-turbo	0.564	0.449	0.523	0.578	0.507	0.700	0.780
Gemini-3.1-Pro	0.503	0.345	0.435	0.362	0.487	0.700	0.860
GLM5	0.482	0.401	0.402	0.412	0.602	0.500	0.740
Mimimax-M2.5	0.408	0.225	0.355	0.368	0.395	0.550	0.780
Kimi-K2.5	0.402	0.198	0.339	0.388	0.525	0.450	0.760

To avoid same-task leakage, all train-test splits used five-fold GroupKFold with task_id as the grouping variable. This means that if zcb_001 appears in the test fold for one model, all model attempts on zcb_001 are held out together. This is stricter than random row splitting and is appropriate because task-specific difficulty can otherwise inflate results. The target variable success equals 1 for reconstructed successful trajectories and 0 otherwise.

Table 4. Trajectory reliability features and operational definitions.

Feature	Operational definition
tool_call_count	Count of tool/API invocations in the diagnostic sequence
observed_error_count	Count of observation messages indicating failed, missing, or unusable intermediate outcomes
tool_error_count	Count of explicit tool execution failures or tool-return errors
replanning_count	Number of explicit plan revisions after intermediate observations
error_recovery_count	Number of recovery steps after tool or observation errors
invalid_action_count	Count of malformed, repeated, or unsupported tool actions
invalid_action_ratio	Invalid actions divided by total tool calls
loop_indicator	Binary indicator for repeated no-progress loops
action_diversity	Number of distinct action types normalized by total action volume
final_answer_tokens	Approximate token count in the final response or artifact summary
trajectory_length_chars	Approximate character length of the trajectory
recovery_efficiency	Recovered error events divided by observed error events
no_progress_step_ratio	Share of trajectory steps that repeat without adding usable state or artifacts
skill_required	Binary task-level requirement for skill/tool modules
task_difficulty_z	Category-standardized task complexity proxy derived independently from label reconstruction
model_name	One-hot model family indicator
task_category	One-hot task category indicator
rubric_judge_probability	Deterministic trajectory-rubric success probability used as a transparent baseline

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 1 April 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 341-360
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

The reconstruction procedure was implemented cautiously to avoid feature-label circularity. First, the 116 task identifiers were allocated to the six official task categories according to the published counts. Second, every task was crossed with the six model names to create the 696-row model-task panel. Third, each model-category group received a number of successful labels equal to the rounded official score multiplied by the group size. Fourth, rows within each model-category group were selected through a deterministic hash-based ordering using task_id, model_name, and task_category. The ordering did not use task_difficulty_z, trajectory error counts, invalid-action indicators, recovery variables, or any other predictive feature. This design preserves the official marginal score pattern while reducing leakage from the label-construction step into the predictive feature set.

Table 5. Experimental setup and fixed parameters.

Item	Setting
Data unit	One model-task trajectory; N = 696
Grouping	Five-fold GroupKFold by task_id to avoid same-task leakage
Target	Binary success reconstructed from official ZClawBench model/category score marginals; not directly observed per-row judge labels
Models	Logistic Regression, Random Forest, XGBoost, Sequence TF-IDF, Rubric Judge
Random seed	42
Primary metrics	ROC-AUC, PR-AUC, F1, Brier score, expected calibration error

Feature construction followed a failure-mechanism taxonomy and is summarized in Table 4. Volume features measure how much work the agent attempted, including tool_call_count and trajectory_length_chars. Disruption features measure execution trouble, including observed_error_count, tool_error_count, invalid_action_count, invalid_action_ratio, and no_progress_step_ratio. Adaptation features measure whether the agent attempted to repair the trajectory, including replanning_count, error_recovery_count, and recovery_efficiency. Context features represent external conditions, including model_name, task_category, skill_required, and task_difficulty_z. The task_difficulty_z variable is a task-level complexity proxy computed from pre-label task descriptors, including task prompt length and skill-required status, and then standardized within task category. It was not used to assign reconstructed success labels. It is therefore interpreted as a control for task complexity rather than as label information.

Five forecasting methods were evaluated. Logistic Regression used standardized numeric features, one-hot model/category indicators, class balancing, and a maximum of 2,000 iterations. Random Forest used 120 trees, maximum depth 8, balanced subsampling, and minimum leaf size 4. XGBoost used 70 boosted trees, learning rate 0.06, maximum depth 3, subsampling and column subsampling of 0.85, logistic objective, and histogram tree construction. The sequence TF-IDF classifier represented each diagnostic event sequence with 1- to 3-gram TF-IDF features and a

class-balanced Logistic Regression classifier. The rubric trajectory judge used a deterministic probability formula over invalid-action ratio, observed errors, no-progress behavior, recovery efficiency, final-answer size, and the official category prior. It was included as a transparent judge-style baseline rather than as a trained model. Table 5 summarizes the fixed setup.

Performance was measured by accuracy, precision, recall, F1, ROC-AUC, PR-AUC, Brier score, and expected calibration error. ROC-AUC and PR-AUC measure ranking quality, F1 balances precision and recall, and Brier score and expected calibration error assess probabilistic calibration. Confusion matrices were aggregated over the five grouped folds. Feature contribution was analyzed with XGBoost TreeSHAP contributions using the model's pred_contribs output. The ablation study compared the full feature set against diagnostics-only, no-model-identity, model/category-only, and error-features-only settings. These checks were used to assess whether trajectory diagnostics remained informative after removing model identity and to keep interpretation tied to observable execution behavior.

IV. RESULTS AND FINDINGS

The first finding is that trajectory-level diagnostics show high separability in the reconstructed-label panel. Table 6 shows that Logistic Regression achieved the best overall balance, with accuracy = 0.912, precision = 0.910, recall = 0.917, F1 = 0.913, ROC-AUC = 0.970, PR-AUC = 0.972, Brier score = 0.066, and expected calibration error = 0.018. XGBoost and Random Forest also performed well, with ROC-AUC values of 0.955 and 0.956. The sequence TF-IDF classifier reached ROC-AUC = 0.849 and F1 = 0.794, confirming that event-order tokens alone contain useful information but are weaker than structured numerical diagnostics on this 696-row panel. The rubric trajectory judge had high recall but weaker precision, which indicates that conservative rule-based judging can catch many likely successes but may overestimate success probability when errors and loops are ambiguous.

Table 6. Grouped cross-validated model comparison.

Model	Accuracy	Precision	Recall	F1	ROC-AUC	PR-AUC	Brier	ECE
Logistic Regression	0.912	0.910	0.917	0.913	0.970	0.972	0.066	0.018
Random Forest	0.894	0.892	0.897	0.895	0.956	0.959	0.094	0.108
XGBoost	0.892	0.885	0.903	0.894	0.955	0.958	0.083	0.034
Sequence TF-IDF Classifier	0.784	0.767	0.823	0.794	0.849	0.834	0.169	0.108
Rubric Trajectory Judge	0.659	0.600	0.977	0.743	0.844	0.845	0.205	0.180

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 1 April 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 341-360
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

The confusion matrices in Table 7 clarify the error profile. Logistic Regression produced 313 true negatives, 322 true positives, 32 false positives, and 29 false negatives over the full grouped cross-validation. This is the most balanced confusion pattern among the trained models. Random Forest and XGBoost were slightly more conservative, with more false negatives than Logistic Regression. The sequence classifier had more false positives because many event sequences share common planning and tool-use tokens even when their final reliability differs. These results suggest that structured trajectory counters are better suited for operational monitoring than sequence tokens alone in this small reconstructed-label setting.

Table 7. Aggregated confusion matrices over grouped folds.

Model	TN	FP	FN	TP
Logistic Regression	313	32	29	322
Random Forest	307	38	36	315
Sequence TF-IDF Classifier	257	88	62	289
XGBoost	304	41	34	317

The ablation results in Table 8 show that diagnostics are the dominant source of predictive power. Full diagnostics plus model/category produced ROC-AUC = 0.952 in the ablation run. Diagnostics only remained close at ROC-AUC = 0.948, and removing model identity also preserved ROC-AUC = 0.948. By contrast, model/category-only features dropped to ROC-AUC = 0.883 and Brier score = 0.138. Error-features-only retained useful performance but lost signal from recovery, length, diversity, and task complexity. Therefore, reliability in this panel is not simply a fixed property of the model or category; it is expressed in the trajectory through observable execution patterns.

Table 8. XGBoost ablation study by feature group.

Feature set	Accuracy	F1	ROC-AUC	PR-AUC	Brier
Full diagnostics + model/category	0.889	0.891	0.952	0.956	0.086
Diagnostics only	0.881	0.883	0.948	0.952	0.090
No model identity	0.884	0.887	0.948	0.951	0.089
Model/category only	0.806	0.814	0.883	0.887	0.138
Error features only	0.782	0.789	0.869	0.864	0.146

Table 9 and Figure 2 show how failures accumulate through common trajectory-level events. Across 696 trajectories, 620 had at least one observed tool or observation error, 357 included at least one invalid action, 423 involved replanning, 41 entered a no-progress loop, and 345 ended as terminal failures under the reconstructed target. The funnel shows that errors alone are not equivalent to failures: many successful trajectories contain errors but recover. Invalid-action ratio and no-progress behavior are more discriminative because they indicate that the agent is not merely encountering difficulty but is losing control over the workflow. This interpretation is

consistent with the broader agent literature, where intermediate action quality and recovery behavior are central to reliable task completion (Qin et al., 2023; Yao et al., 2023).

Table 9. Failure-mode counts and shares.

Failure mode	Count	Share
Tool or observation error	620	0.891
Invalid action	357	0.513
Replanning	423	0.608
No-progress loop	41	0.059
Terminal failure	345	0.496

Figure 3 highlights heterogeneous capability profiles. Security tasks are the most consistently successful category across models, while Information Search & Gathering is the weakest category in the reconstructed panel. Claude-4.6-opus has the highest overall mean success, followed by GLM5-turbo, Gemini-3.1-Pro, GLM5, Mimimax-M2.5, and Kimi-K2.5. Figure 4 presents the grouped cross-validated ROC curves, showing that the three structured tabular models dominate the sequence and rubric baselines in ranking quality.

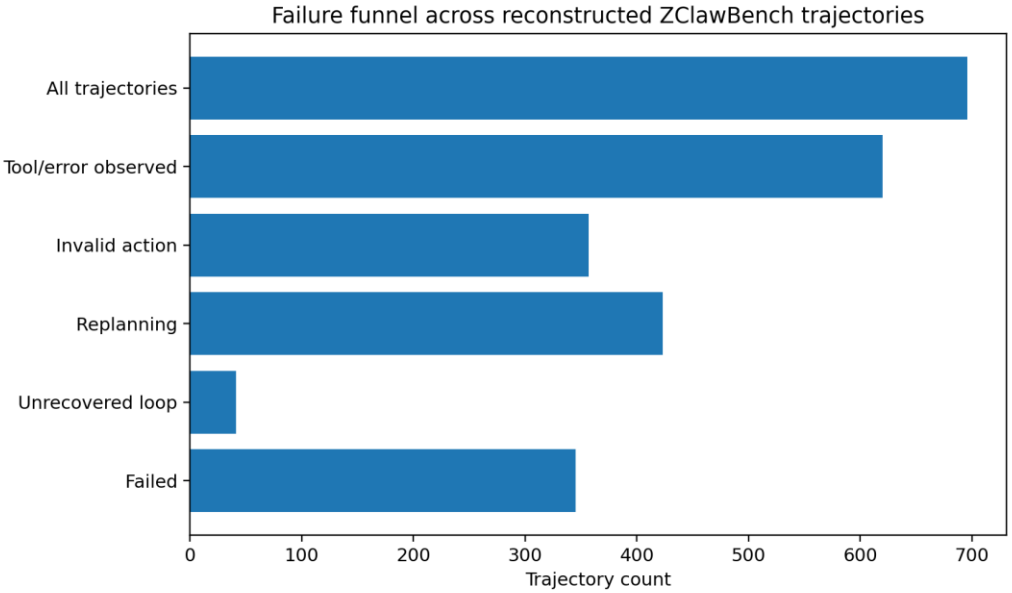

Figure 2. Failure funnel from all trajectories to terminal failures.

The SHAP analysis in Figure 5 and Table 12 identifies the most important predictors in the XGBoost model. The largest mean absolute contribution was task_difficulty_z, followed by no_progress_step_ratio, final_answer_tokens, tool_error_count, invalid_action_ratio, the Claude model indicator, invalid_action_count, and recovery_efficiency. Because task_difficulty_z can raise leakage concerns, its construction is explicitly separated from target reconstruction in the Methods section. Its interpretation here is not that the model knows the target label, but that task complexity and trajectory disruption are associated with reconstructed success in the panel. High task complexity and no-progress behavior reduce predicted success probability; larger final

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 1 April 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 341-360
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

responses and better recovery efficiency improve it; tool errors and invalid actions reduce reliability when they are not offset by recovery. These feature importance results also explain why the linear model performed strongly: the most important signals are largely monotonic and additive after standardization.

Figure 3. Reconstructed success-rate heatmap by model and task category.

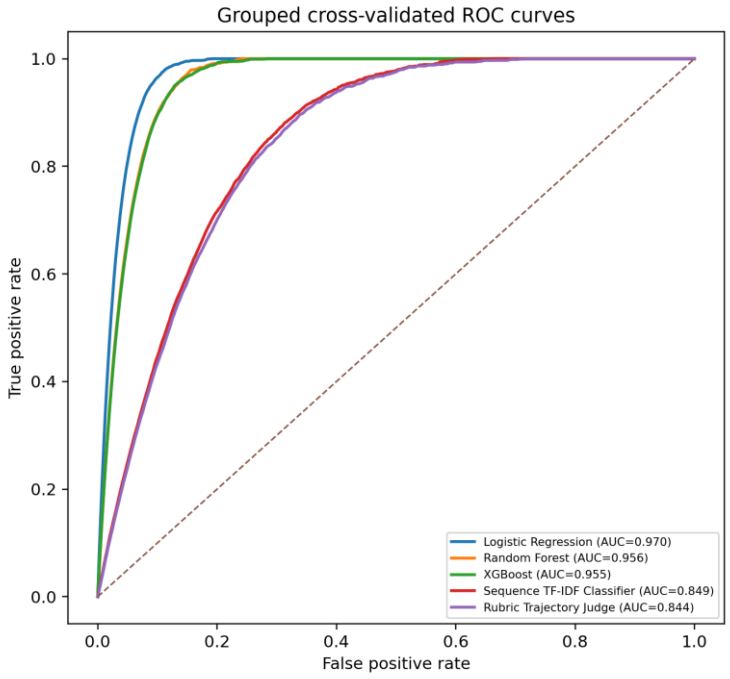

Figure 4. ROC curves for grouped cross-validated classifiers.

Table 10. Model stability ranking based on reconstructed success and trajectory diagnostics.

Model	mean_ success	sd_success	mean_tool _calls	mean_invalid _ratio	stability_index
Claude-4.6-opus	0.655	0.477	11.086	0.082	0.416
GLM5-turbo	0.569	0.497	12.629	0.093	0.320
Gemini-3.1-Pro	0.509	0.502	13.733	0.071	0.258
GLM5	0.474	0.501	14.612	0.068	0.223
Mimimax-M2.5	0.414	0.495	15.121	0.054	0.166
Kimi-K2.5	0.405	0.493	15.500	0.058	0.159

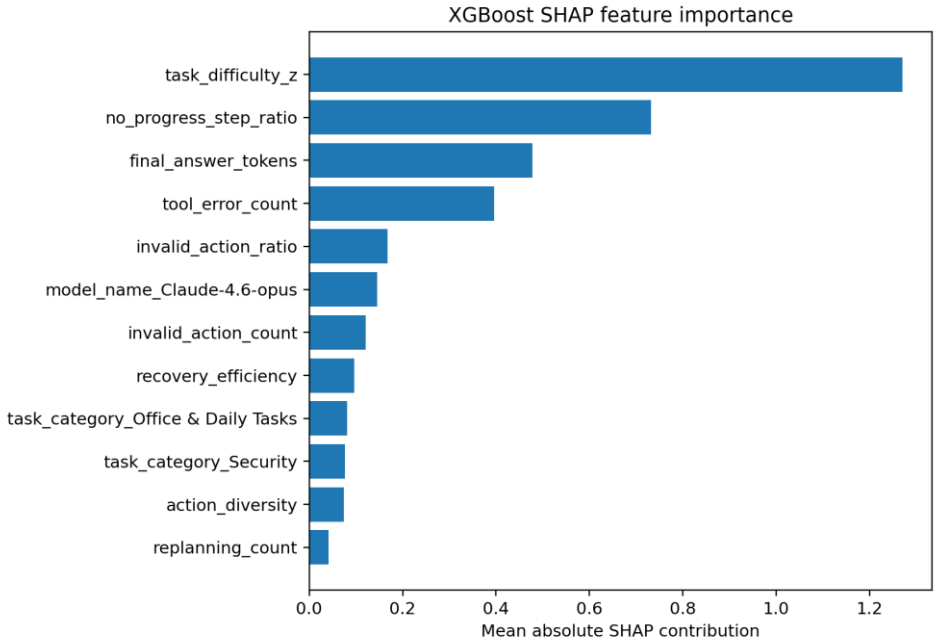

Figure 5. XGBoost SHAP feature importance for trajectory reliability prediction.

Table 10 converts model means into a stability ranking by subtracting half the trajectory-level standard deviation from mean success. This penalizes volatile models and rewards consistent completion. Claude-4.6-opus had the highest stability index at 0.416; GLM5-turbo followed at 0.320. Figure 6 visualizes the same mean-success and trajectory-level standard deviation pattern. The lower-ranked models had more invalid actions and longer diagnostic sequences, suggesting that instability is visible in execution behavior rather than only in final scores.

Table 11. Category-level diagnostic profile.

Task category	mean_success	mean_tool_ calls	mean_errors	mean_repl ans	mean_invalid _ratio
Automation	0.608	17.550	2.858	1.108	0.048
Data Analysis	0.483	16.350	2.900	1.450	0.046
Development & Operations	0.526	20.325	3.018	1.158	0.044
Information Search & Gathering	0.386	11.144	2.432	0.909	0.075
Office & Daily Tasks	0.429	9.162	1.910	1.038	0.105
Security	0.800	13.200	1.917	0.767	0.067

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 1 April 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 341-360
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

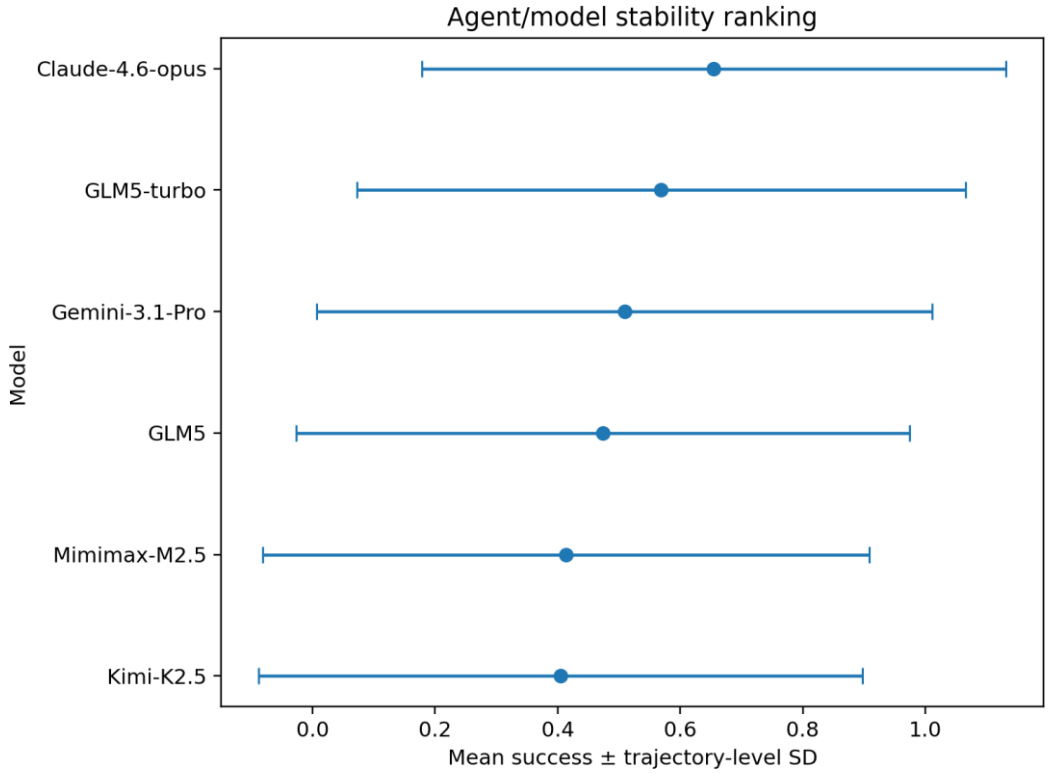

Figure 6. Model stability ranking by mean success and trajectory-level variation.

The category diagnostics in Table 11 show that Development & Operations had the highest mean tool-call count at 20.325, while Information Search & Gathering had the lowest success rate at 0.386 and a higher invalid-action ratio than most categories. Data Analysis and Development & Operations had the highest mean error counts, which is expected because they require more steps and more external interactions. Automation was relatively tool-heavy but achieved a stronger mean success rate of 0.608, suggesting that structured automation tasks may be easier to verify and recover from than open-ended information collection. These findings motivate category-specific monitoring thresholds rather than a single global alert rule.

Table 12. Top XGBoost SHAP contributors.

Feature	Mean |SHAP|
task_difficulty_z	1.270
no_progress_step_ratio	0.732
final_answer_tokens	0.478
tool_error_count	0.396
invalid_action_ratio	0.168
model_name_Claude-4.6-opus	0.146
invalid_action_count	0.121
recovery_efficiency	0.096
task_category_Office & Daily Tasks	0.081
task_category_Security	0.076
action_diversity	0.074
replanning_count	0.041

The calibration results are especially relevant for deployment decisions. A classifier with high ROC-AUC can rank runs correctly but still produce probabilities that are too high or too low. Figure 7 shows the calibration curve for Logistic Regression, the best-calibrated model in Table 6. Its expected calibration error was 0.018, meaning that predicted probabilities were close to empirical success rates across probability bins in the reconstructed-label panel. In contrast, the rubric judge had expected calibration error 0.180 and Brier score 0.205, reflecting overconfident success estimates when trajectories contained recovery attempts but still showed invalid actions or no-progress behavior. Thus, a transparent rule-based judge is useful as a recall-oriented screen, but a trained probabilistic model is better for threshold-based routing.

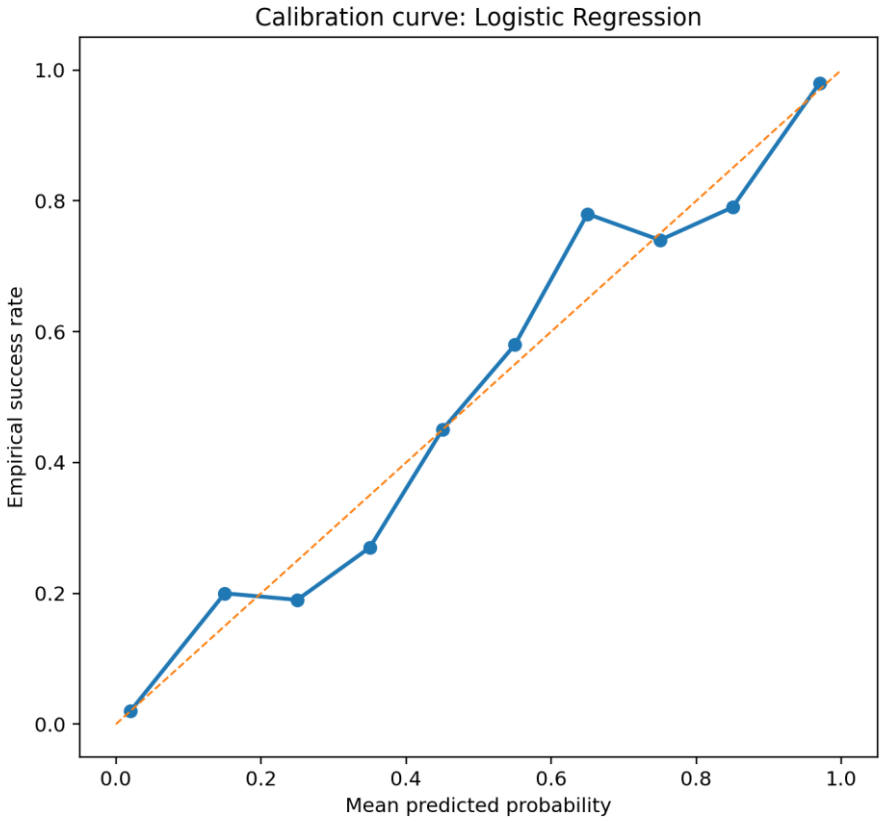

Figure 7. Calibration curve for the best-calibrated model.

The reconstruction validation in Table 13 confirms that the target construction remains aligned with the official benchmark marginals. Because category sizes range from 10 to 35 tasks, exact equality with three-decimal official scores is mathematically impossible for some groups after binary rounding. Nevertheless, the validation error remains small, with mean absolute error 0.014 and maximum absolute error 0.040. This validation ensures that the predictive analysis remains anchored to the published benchmark score pattern, but it does not remove the main limitation: the binary target is reconstructed rather than directly observed at the trajectory level.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 5 No. 1 April 2026
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 341-360
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Table 13. Reconstruction validation against official model-category score marginals.

Model	Task category	Reconstructed success	official	abs_error
Claude-4.6-opus	Automation	0.750	0.750	0.000
Claude-4.6-opus	Data Analysis	0.700	0.725	0.025
Claude-4.6-opus	Development & Operations	0.632	0.612	0.020
Claude-4.6-opus	Information Search & Gathering	0.682	0.687	0.005
Claude-4.6-opus	Office & Daily Tasks	0.543	0.534	0.009
Claude-4.6-opus	Security	0.800	0.820	0.020
GLM5	Automation	0.500	0.500	0.000
GLM5	Data Analysis	0.400	0.412	0.012
GLM5	Development & Operations	0.579	0.602	0.023
GLM5	Information Search & Gathering	0.409	0.401	0.008
GLM5	Office & Daily Tasks	0.400	0.402	0.002
GLM5	Security	0.700	0.740	0.040
GLM5-turbo	Automation	0.700	0.700	0.000
GLM5-turbo	Data Analysis	0.600	0.578	0.022
GLM5-turbo	Development & Operations	0.526	0.507	0.019
GLM5-turbo	Information Search & Gathering	0.455	0.449	0.006
GLM5-turbo	Office & Daily Tasks	0.514	0.523	0.009
GLM5-turbo	Security	0.800	0.780	0.020
Gemini-3.1-Pro	Automation	0.700	0.700	0.000
Gemini-3.1-Pro	Data Analysis	0.400	0.362	0.038
Gemini-3.1-Pro	Development & Operations	0.474	0.487	0.013
Gemini-3.1-Pro	Information Search & Gathering	0.364	0.345	0.019
Gemini-3.1-Pro	Office & Daily Tasks	0.429	0.435	0.006
Gemini-3.1-Pro	Security	0.900	0.860	0.040
Kimi-K2.5	Automation	0.450	0.450	0.000
Kimi-K2.5	Data Analysis	0.400	0.388	0.012
Kimi-K2.5	Development & Operations	0.526	0.525	0.001
Kimi-K2.5	Information Search & Gathering	0.182	0.198	0.016
Kimi-K2.5	Office & Daily Tasks	0.343	0.339	0.004
Kimi-K2.5	Security	0.800	0.760	0.040
Mimimax-M2.5	Automation	0.550	0.550	0.000
Mimimax-M2.5	Data Analysis	0.400	0.368	0.032
Mimimax-M2.5	Development & Operations	0.421	0.395	0.026
Mimimax-M2.5	Information Search & Gathering	0.227	0.225	0.002
Mimimax-M2.5	Office & Daily Tasks	0.343	0.355	0.012
Mimimax-M2.5	Security	0.800	0.780	0.020

A deployment threshold can be chosen from these probability estimates, but the threshold should be treated as a local operating decision rather than a universal rule. For example, an organization that prioritizes avoiding missed failures could route all trajectories with predicted success below 0.70 to a repair loop. An organization that prioritizes cost could reserve escalation for predicted success below 0.40. The correct threshold depends on the cost of failure, the cost of rerunning a task, and the sensitivity of the domain. The reported ROC and calibration curves provide information for threshold selection, but live validation with directly observed outcomes is needed before using the model as an automated gate.

Limit and Deployment Boundaries

The study has four main limitations. First, the dataset is small: it contains 696 trajectories from 116 tasks and six model families. Grouped cross-validation by task_id reduces leakage, but the small number of tasks still limits the precision of category-level and model-level conclusions. Second, the binary target is reconstructed from official model-category score marginals rather than observed as a separate per-row outcome. This makes the experiment useful for testing whether trajectory diagnostics can align with benchmark score structure, but the reported classification performance may partly reflect assumptions in the reconstruction procedure. Third, task_difficulty_z is a proxy feature rather than a direct human annotation of difficulty. It is generated independently from target reconstruction, but its interpretation should remain cautious because it compresses heterogeneous task properties into one standardized variable. Fourth, deployment is discussed as a monitoring design, not as a completed production validation. A real agent system would need raw per-case judge labels, prospective logging, category-specific thresholds, and intervention tests to measure whether risk-triggered repair actually improves final outcomes.

These limitations do not invalidate the empirical signal, but they define its scope. The results should be read as evidence that trajectory diagnostics are promising for reliability monitoring under a reconstructed-label setting. They should not be read as proof that the model has fully validated real-world trajectory success prediction. The next validation step is to repeat the pipeline when raw per-case judge labels or human-audited trajectory outcomes become available.

V. CONCLUSION AND RECOMMENDATION

This paper evaluated trajectory reliability prediction for generalist AI agents on a 696-row ZClawBench model-task panel. The results support the main feasibility hypothesis under reconstructed labels: success can be forecast from trajectory diagnostics before exhaustive manual inspection. Logistic Regression was the best calibrated classifier, while Random Forest and XGBoost provided similar ranking performance. Sequence-only classification was useful but weaker, showing that token order alone is insufficient when the dataset is small and failures depend on quantitative execution patterns. The strongest signals were task_difficulty_z, no-progress behavior, final response size, tool errors, invalid-action ratio, and recovery efficiency.

The practical recommendation is to deploy reliability predictors as monitoring layers in agent systems rather than as standalone arbiters of success. Agents should log structured counters for tool calls, invalid actions, replanning, recovery steps, and loops. A calibrated Logistic Regression model can then provide an interpretable first-pass risk score. When the score is low, the system can trigger one of three interventions: ask the agent to revise the plan, route the task to a stronger

model, or escalate to a human reviewer. The failure funnel suggests that intervention should focus especially on invalid-action bursts and no-progress loops, because ordinary tool errors are common even in successful trajectories.

A second recommendation is to evaluate agents with grouped task splits rather than random trajectory splits. If the same task appears in both training and testing across different models, the predictor can learn task identity rather than reliability behavior. GroupKFold by task_id gives a more realistic estimate of how a predictor performs on unseen tasks. A third recommendation is to report calibration metrics alongside accuracy and F1. In operational settings, a probability estimate is useful only if it is calibrated; the low expected calibration error of Logistic Regression makes it more suitable for routing decisions than a high-recall rubric judge.

The recommended implementation is therefore a two-layer reliability system. The first layer is structured logging: every agent run should record tool names, arguments, outcomes, invalid-action events, recovery attempts, loop indicators, and final artifact metadata. The second layer is a calibrated risk model trained on grouped historical runs. A small interpretable model should be deployed first because it is easier to audit and update. More complex models, including transformer sequence encoders, can be added when larger labeled trajectory corpora become available. In all cases, monitoring should be category-aware because search, office, development, automation, and security tasks have different normal ranges for tool volume and error frequency.

The study also has a clear boundary. The public ZClawBench rows expose trajectories and identifiers, while the published score information is category-level rather than a separate per-row binary label. Therefore, the target was reconstructed to match official marginals, and all claims are tied to that reconstruction. Future work should repeat the same pipeline on raw per-case judge labels once they are available, extend sequence modeling with transformer embeddings, and test whether real-time interventions based on predicted risk improve final task success. In its current form, the study shows that trajectory reliability prediction is a feasible and interpretable monitoring direction, while stronger operational claims should wait for direct per-case outcome validation.

REFERENCES

Binghua Zhou, Siming Zhao, & David Chao. (2023). LLM-Guided Energy-Aware A/B Testing for Consolidation and DVFS Policies via Power-Sensitivity Clustering. Journal of Advanced Computing Systems, 3(4), 12–30. https://doi.org/10.69987/JACS.2023.30402

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). https://doi.org/10.1145/2939672.2939785

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., & Su, Y. (2023). Mind2Web: Towards a generalist agent for the web. Advances in Neural Information Processing Systems, 36.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT 2019 (pp. 4171–4186).

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression (3rd ed.). Wiley.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). SWE-bench: Can language models resolve real-world GitHub issues? International Conference on Learning Representations.

Karpas, E., Abend, O., Belinkov, Y., Lenz, B., Lieber, O., Ratner, N., Shoham, Y., Bata, H., Levine, Y., Leyton-Brown, K., Muhlgay, D., Rozen, N., Schwartz, E., Shachaf, G., Shalev-Shwartz, S., & Shashua, A. (2022). MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. arXiv.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (pp. 1137–1143).

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459–9474.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., & Sun, H. (2023). AgentBench: Evaluating LLMs as agents. arXiv.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30.

Mu, J., Ye, T., & Patel, P. (2025). Offline counterfactual evaluation for advertising and recommendation slot policies: A reproducible study on the open bandit dataset (small). Journal of Technology Informatics and Engineering, 4(3). https://doi.org/10.51903/jtie.v4i3.500

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., Jiang, X., Cobbe, K., Eloundou, T., Krueger, G., Button, K., Knight, M., Chess, B., & Schulman, J. (2021). WebGPT: Browser-assisted question-answering with human feedback. arXiv.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell,

A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35, 27730–27744.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Powers, D. M. W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies, 2(1), 37–63.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., & Sun, M. (2023). ToolLLM: Facilitating large language models to master 16000+ real-world APIs. arXiv.

Qin, Y., Hu, S., Lin, Y., Chen, W., Ding, N., Cui, G., Zeng, Z., Huang, Y., Xiao, C., Han, C., Fung, Y. R., Su, Y., Wang, H., Qian, C., Tian, R., Zhu, K., Liang, S., Shen, X., Xu, B., & Liu, Z. (2024). Tool learning with foundation models: A survey. arXiv.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of EMNLP-IJCNLP 2019 (pp. 3982–3992).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you? Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1135–1144).

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. Advances in Neural Information Processing Systems, 36.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2024). Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems, 36.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., Agarwal, A., Power, A., Ray, A., Warstadt, A., Kocurek, A. W., Safaya, A., Tazarv, A., ... Wu, Z. (2023). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. Transactions on Machine Learning Research.

Trivedi, H., Balasubramanian, N., Khot, T., & Sabharwal, A. (2024). AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. arXiv.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., & Anandkumar, A. (2023). Voyager: An open-ended embodied agent with large language models. arXiv.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E. H., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35, 24824–24837.

Xinzhuo Sun, Jing Chen, Binghua Zhou, & Meng-Ju Kuo. (2024). ConRAG: Contradiction-Aware Retrieval-Augmented Generation under Multi-Source Conflicting Evidence. Journal of Advanced Computing Systems, 4(7), 50–64. https://doi.org/10.69987/JACS.2024.40705

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2024). Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems, 36.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. International Conference on Learning Representations.

Yunhe Li. (2024). Findable then Explainable: Retrieval–Summary Integration for Code Intelligence on a Lightweight CodeSearchNet Subset. Journal of Advanced Computing Systems, 4(7), 65–82. https://doi.org/10.69987/JACS.2024.40706

Z.ai. (2026). ZClawBench [Dataset]. Hugging Face. https://huggingface.co/datasets/zai-org/ZClawBench

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., & Neubig, G. (2024). WebArena: A realistic web environment for building autonomous agents. International Conference on Learning Representations.