LITMUS (RE)AGENT: A Benchmark and Agentic System for Predictive Evaluation of Multilingual Models

Anonymous ACL submission

Abstract

We study *predictive multilingual evaluation*: estimating how well a model will perform on a task in a target language when direct benchmark results are missing. This problem is common in multilingual deployment, where evaluation coverage is sparse and published evidence is uneven across languages, tasks, and model families. We introduce a controlled benchmark of 1,500 questions spanning six tasks and five evidence scenarios. The benchmark separates *accessible evidence* from *ground truth*, enabling evaluation of systems that must infer missing results from incomplete literature evidence. We also present LITMUS (RE)AGENT, a DAG-orchestrated agentic system that decomposes queries into hypotheses, retrieves evidence, and synthesises predictions through feature-aware aggregation. Across six systems, LITMUS (RE)AGENT achieves the best overall performance, with the largest gains in transfer-heavy scenarios where direct evidence is weak or absent. These results show that structured agentic reasoning is a promising approach to multilingual performance estimation under incomplete evidence.

1 Introduction

Large Language Models (LLMs) are increasingly expected to operate across many tasks and many languages, yet multilingual evaluation remains highly incomplete (OpenAI, 2023; NLLB Team, 2022). For many task–model–language combinations, especially in low-resource settings, direct benchmark results are unavailable, scattered across papers, reported under inconsistent conditions, or too expensive to reproduce. As a result, practitioners often need to make deployment and model-selection decisions without the exact evidence they would ideally want. This creates a practical question: *how will a model perform on a task in a target language when direct evaluation results are missing?* We refer to this as the *Task–Model–Language* (TML) prediction problem.

Existing approaches address this problem only partially. Multilingual benchmark suites expand evaluation coverage, but still leave large portions of the TML space unobserved (Ruder et al., 2021; Liang et al., 2022; NLLB Team, 2022). Predictive transfer methods use typological, representational, or information-theoretic signals (Lin et al., 2019; Nguyen et al., 2020; Srinivasan et al., 2022; Tsvetkov and Kipnis, 2024), but usually rely on fixed features rather than reasoning over scientific evidence. LLM-as-judge methods scale evaluation, but they do not directly solve missing-result prediction from incomplete literature evidence and also raise concerns about bias and reproducibility (Tan et al., 2024). More recent agentic systems can retrieve and synthesise information from research papers (Mittal et al., 2025), but predictive reasoning under systematically restricted evidence remains underexplored. The field therefore still lacks a controlled framework for studying how systems infer missing multilingual performance from incomplete literature evidence.

To address this gap, we introduce both a benchmark and a system. First, we construct a controlled benchmark for predictive multilingual evaluation. The benchmark contains 1,500 questions spanning six tasks and five evidence scenarios, and separates *accessible evidence* at inference time from *ground truth*. Systems are evaluated using only a reduced paper corpus, while answers are defined from a larger combined corpus. This enables controlled study of predictive reasoning under varying evidence conditions. The benchmark covers two complementary capabilities: numeric performance prediction and comparative multilingual reasoning.

Second, we present LITMUS (RE)AGENT, a DAG-orchestrated agentic system for this setting. The system decomposes a query into hypotheses, retrieves citation-grounded evidence, extracts lin-

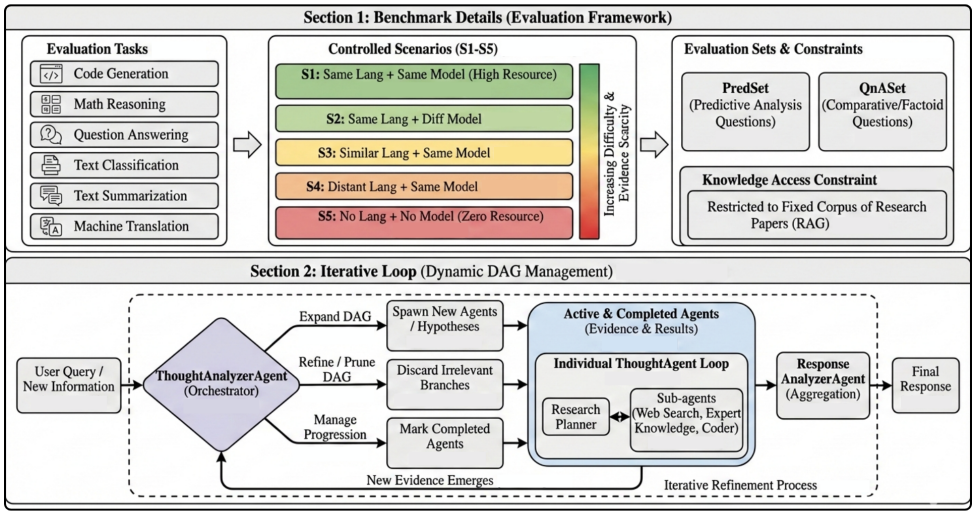


Figure 1: Overview of the benchmark and LITMUS (RE)AGENT. Top: six tasks, five controlled scenarios (S1–S5), two query types, and restricted paper-corpus access. Bottom: dynamic DAG orchestration in which specialised agents spawn and prune hypotheses, gather evidence, and aggregate results into the final response.

guistic and task-level signals, and aggregates them into a final prediction. Relative to the earlier DAG-based system of Mittal et al. (2025), our version strengthens expert-knowledge retrieval, expands coder support with linguistic feature libraries, and improves prompting for more stable, expert-aligned reasoning.

We evaluate LITMUS (RE)AGENT against five baselines, including the prior DAG-based system, single-agent and non-DAG agentic variants, a direct GPT-4.1 baseline, and a general-purpose multi-agent framework. Across the full benchmark, LITMUS (RE)AGENT achieves the strongest overall performance, with the largest gains in transfer-heavy settings where direct evidence is weak or absent. We further provide analysis across tasks, evidence scenarios, metric types, and internal agent behaviours, together with a human evaluation study.

Our contributions are threefold: (i) we introduce a controlled benchmark for predictive multilingual evaluation under incomplete evidence, spanning six tasks, five evidence scenarios, and both numeric prediction and comparative reasoning; (ii) we present LITMUS (RE)AGENT, a DAG-orchestrated, citation-grounded system for estimating multilingual performance from incomplete literature evidence; and (iii) we provide a comprehensive empirical analysis, including comparisons against five baselines, breakdowns by task and evidence scenario, internal agent-behaviour diagnostics, and a human evaluation study.

2 Related Work

2.1 Multilingual Evaluation and Benchmarks

Multilingual evaluation has been studied through benchmark suites and task-specific datasets. Resources such as XTREME, XTREME-R, and XGLUE evaluate cross-lingual transfer across multiple languages and tasks (Hu et al., 2020; Ruder et al., 2021; Liang et al., 2020), while broader platforms such as HELM and BIG-Bench extend evaluation across diverse capabilities and models (Liang et al., 2022; Srivastava et al., 2023). Task-specific benchmarks further cover program synthesis, mathematical reasoning, and summarisation (Chen et al., 2021; Cobbe et al., 2021; Narayan et al., 2018). These resources provide valuable multilingual evidence, but they do not address the problem of predicting missing results under incomplete coverage.

2.2 Predictive Multilingual Analysis

A separate line of work estimates performance in unseen language–task settings without running full evaluations. Early approaches used typological and corpus-based signals such as LangRank (Lin et al., 2019), followed by representation-based, information-theoretic, and transferability methods including Task2Vec, LEEP, and LogME (Achille et al., 2019; Nguyen et al., 2020; You et al., 2021). Later work introduced meta-predictors, ranking-based approaches, and multilingual deployment frameworks such as LITMUS (Ahuja

et al., 2022a; Srinivasan et al., 2022), while recent methods explore scalable prediction through information-theoretic modelling and language ranking (Tsvetkov and Kipnis, 2024; Schram et al., 2023; Li et al., 2025). These approaches are closely related to our setting, but they generally rely on predefined features rather than literature-grounded reasoning under controlled evidence restriction.

2.3 Agentic and Literature-Grounded Systems

Agentic systems augment language models with planning, tool use, and structured reasoning. ReAct and Toolformer introduced interleaved reasoning and tool invocation (Yao et al., 2022; Schick et al., 2023), while graph- and multi-agent frameworks support more structured workflows (Zhuge et al., 2024; Liu et al., 2025). Retrieval-augmented systems and scientific assistants, including PaperQA and related research agents, support citation-grounded reasoning over documents (Lewis et al., 2020; Liu et al., 2023; Zhang et al., 2024a). Most of this work focuses on literature exploration or question answering rather than multilingual performance prediction. The closest prior work is Mittal et al. (2025), which introduced a DAG-based predictive analysis system but did not provide a large, scenario-controlled benchmark for studying performance under systematically incomplete evidence.

3 Benchmark Details

We construct a controlled benchmark for multilingual predictive evaluation under incomplete evidence. Each instance corresponds to a Task–Model–Language query and is assigned to one of five scenarios that vary the evidence available at inference time. The key design is to separate *accessible evidence* from *ground truth*: systems retrieve from a reduced paper corpus, while answers are defined from a larger combined corpus.

3.1 Benchmark Scope

The benchmark spans six tasks: code generation, mathematical reasoning, question answering (QA/VQA), text classification (including NLI), text summarisation, and machine translation. We curate multilingual evaluation papers and extract language-to-model-family mappings, which we aggregate into a task-specific combined mapping that represents the full answer space. To simulate incomplete evidence, we construct a reduced mapping by removing a subset of papers from the combined corpus. Systems access only the reduced corpus during evaluation, while ground-truth values are taken from the combined set, enabling controlled analysis under partial evidence.

3.2 Query Types: Numeric Prediction and Comparative Reasoning

The benchmark evaluates two complementary capabilities: numeric performance prediction and comparative multilingual reasoning. PREDSET requires predicting a scalar score for a given task, model family, and language, while QNASET requires comparing candidate models or languages to identify the best-performing option. This distinction reflects practical usage: users ask both *"What score should I expect?"* and *"Which model is likely to be best?"*, so the benchmark covers both regression and comparative reasoning.

Numeric Prediction (Code Generation):
Q1: *What is the performance of DeepSeek-Coder-V2 on Code Generation for Nepali?*
Answer: 1.5 (pass@1)

Numeric Prediction (Classification/NLI):
Q2: *What score does GPT-4-Turbo achieve for Classification NLI in Oromo?*
Answer: 10.7

Numeric Prediction (Machine Translation):
Q3: *How well does IndicTrans2 perform on Machine Translation for Sanskrit?*
Answer: 51.6 (chrF++)

Comparative Reasoning (Text Summarisation):
Q4: *Which model achieves the highest score for Text Summarization in Kirundi?*
Answer: Qwen2-7B

Comparative Reasoning (QA/VQA):
Q5: *What is the top-performing model for QA VQA in Tigrinya?*
Answer: TiRoBERTa

Figure 2: Illustrative questions from the two benchmark subsets: PREDSET for numeric prediction and QNASET for comparative reasoning.

3.3 Scenario Design

Table 1 summarises the five evidence scenarios used in the benchmark. Throughout, **same model** refers to the same **model family**, rather than the exact checkpoint identifier. **S1** is the strongest-evidence setting, where both the target language and target model family are observed in the reduced corpus, allowing direct retrieval for the exact Task–Model–Language combination. **S2** keeps the target language observed but removes the target model family, so prediction must transfer across models within the same language. **S3** and **S4** both remove

Scenario	Target language	Target model	Available evidence source
S1: Same L + Same M	Observed	Observed	Direct evidence
S2: Same L + Diff M	Observed	Missing	Same language, other models
S3: Similar L + Same M	Missing	Observed	Typologically close language
S4: Distant L + Same M	Missing	Observed	Typologically distant language
S5: Diff L + Diff M	Missing	Missing	Other languages and model families

Table 1: Summary of the five benchmark scenarios. "Observed" means directly available in the reduced evidence corpus at inference time.

the target language while retaining evidence for the same model family: **S3** allows transfer from a typologically close language, whereas **S4** allows only distant-language evidence. **S5** is the weakest-evidence setting, where both the target language and target model family are absent, requiring transfer across both languages and model families.

3.4 Question Construction

Questions are generated for each task–scenario block from language-to-model-family mappings. Prompts are constrained to use only languages and model families present in the mapping, and exclude programming languages for code generation. Each block includes both numeric-prediction and comparative-reasoning questions. The full prompt template is provided in Appendix H (Figure 21).

3.5 Ground Truth and Metric Normalisation

Ground-truth values are automatically extracted from the paper collection and manually validated. We record the associated metric for each question (e.g., *pass@1*, Accuracy, F1, BLEU) and normalise all scores to a common 0–100 scale. Values in $[0, 1]$ are multiplied by 100, while other metrics undergo task-specific linear scaling. Evaluation is restricted to compatible metric families within each task to ensure valid comparisons.

3.6 Language Similarity for Transfer Scenarios

For Scenarios 3 and 4, language similarity is defined using cosine distance over `lang2vec` features, with language pairs split into *close* and *distant* groups via a percentile-based threshold. Full details are provided in Appendix H.4.

3.7 Dataset Statistics

The benchmark contains 1,500 questions across six tasks and five scenarios, with 50 questions per task–scenario block (25 PREDSET and 25 QNASET). The reduced evidence set is constructed to ensure

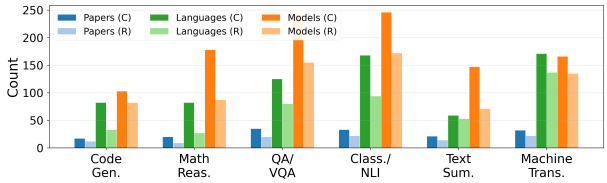


Figure 3: Combined versus reduced corpus statistics per task.

all scenarios remain instantiable for every task. Figure 3 compares the combined and reduced paper collections. The reduced set has lower language and model-family coverage, creating controlled evidence scarcity while retaining ground truth in the combined set. Additional statistics are provided in Appendix H.

4 LITMUS (RE)AGENT

We build on a DAG-based multi-agent architecture originally proposed for multilingual performance prediction (Mittal et al., 2025). That system showed that decomposing evaluation queries into parallel hypothesis threads handled by specialised *Thought Agents* yields more grounded predictions than monolithic prompting. We retain the core DAG structure but introduce targeted improvements to expert-knowledge retrieval, code execution with linguistic features, and prompt design. These modifications produce LITMUS (RE)AGENT, enabling reliable predictive analysis under restricted evidence by aligning system behaviour with expert multilingual evaluation workflows. Implementation details are provided in Appendix C.

4.1 Base Architecture

The system follows a DAG-based orchestration pattern (Zhang et al., 2024b). A natural-language query (e.g., *"How will GPT-4o perform on machine translation in Nepali?"*) is processed by a *MainAgent*, which determines whether it is new or a follow-up. For new queries, a *ThoughtCreator-*

Agent decomposes the problem into hypotheses and spawns one *ThoughtAgent* per hypothesis as a node in the DAG. Each ThoughtAgent operates as a coordinated group of specialised sub-agents: **Research Planner** (plans evidence acquisition and coordinates analysis), **Web Search and Crawl** (retrieves and parses documents), **Expert Knowledge** (queries a curated multilingual knowledge base), **Coder** (writes and executes analysis scripts, including regression models), and **Reporter** (summarises the hypothesis outcome).

A *ThoughtAnalyzerAgent* monitors active hypotheses, spawning new ones when gaps appear and pruning redundant branches. Once all hypotheses are resolved, a *ResponseAnalyzerAgent* aggregates the results into a final response containing the predicted outcome, supporting citations, and a structured rationale. This modular design limits context-window saturation while preserving interpretable reasoning, as each hypothesis maintains an independent evidence trail.

4.2 Improvements in LITMUS (RE)AGENT

Relative to the base system, we introduce three improvements that enhance planning, retrieval, and validation in line with expert workflows.

Enriched Expert Knowledge. The base system relied on a small set of task-specific papers. We expand this to a broader corpus of multi-task cross-lingual evaluation studies, reflecting the transferability of predictive signals (Ahuja et al., 2022b; Srinivasan et al., 2022). The knowledge base encodes multilingual evaluation strategies, including language–task–model observations, failure modes, and expert analysis patterns, enabling more informed prediction planning.

Enhanced Code Execution with Linguistic Feature Libraries. The Coder agent previously had limited library access and lacked structured linguistic resources. We extend it with `lang2vec` and the URIEL typological database (Littell et al., 2017), providing syntactic, phonological, geographic, and family-level features for over 7,000 languages. These enable language distance computation, feature extraction, and feature-informed regression models used in multilingual research.

Prompt Stabilisation and Expert-Aligned Reasoning. Prompts are redesigned to reinforce the analysis process and effective use of system capabilities. Instructions guide agents to follow expert-style workflows, including hypothesis generation, evidence retrieval, and feature-based analysis, while requiring cited evidence and careful tool usage. This reduces failure modes such as invalid tool calls, brittle code execution, and hallucinated citations.

4.3 Evaluation Setup

We evaluate all methods on the benchmark described in Section 3. During evaluation, retrieval is restricted to the *reduced* paper corpus, while ground truth is computed from the *combined* corpus (Section 3.5). This restriction is part of the evaluation protocol rather than a limitation of the system itself: it prevents direct access to held-out results and enables controlled study of predictive reasoning under incomplete evidence.

We compare LITMUS (RE)AGENT against five baselines: (i) **LITMUS++**, the prior DAG-based system without the improvements introduced in this work; (ii) **ThoughtAgent**, a single multi-agent group chat without DAG decomposition; (iii) **Single Agent**, a ReAct-style agent with direct tool access but no multi-agent coordination; (iv) a **GPT-4.1 direct baseline** with no agentic scaffolding; and (v) **Magentic-One** (Fourney et al., 2024), a general-purpose multi-agent framework. All agentic systems use GPT-4.1 as the underlying backbone. Additional implementation details are provided in Appendix B.

5 Results

We evaluate LITMUS (RE)AGENT on the full 1,500-question benchmark and compare it against five baselines: LITMUS++, ThoughtAgent, Single Agent, GPT-4.1 direct, and Magentic-One. All agentic systems use GPT-4.1 as the backbone LLM.

5.1 Overall Performance

Tables 2 summarise the main findings. On PRESET, LITMUS (RE)AGENT achieves an overall MAE of **10.3**, followed by LITMUS++ (12.7), Single Agent (14.5), ThoughtAgent (15.1), GPT-4.1 (16.4), and Magentic-One (17.2). On QNASET, LITMUS (RE)AGENT reaches **21.6%** accuracy, compared to 18.9% for the Single Agent, 15.9% for both ThoughtAgent and GPT-4.1, 15.1% for LITMUS++, and 10.0% for Magentic-One. These results show that LITMUS (RE)AGENT improves both numeric prediction and comparative reasoning over all baselines, and that the three targeted

	PredSet MAE (↓)						QnASet Accuracy (%) (↑)					
Task	**LRA**	**L++**	**TA**	**SA**	**GPT**	**M1**	**LRA**	**L++**	**TA**	**SA**	**GPT**	**M1**
Code Gen.	**14.2**	21.1	18.9	17.8	39.8	27.0	**55.3**	25.4	39.5	51.8	27.2	41.2
Math Reas.	20.0	28.8	27.5	29.6	**18.2**	26.8	15.4	10.6	13.0	9.8	**18.7**	7.3
QA/VQA	9.5	**8.7**	14.7	12.4	9.6	14.1	8.0	**22.4**	8.0	16.8	16.0	5.6
Class./NLI	**7.9**	9.6	11.6	11.9	9.9	12.0	**16.0**	7.2	6.4	13.6	9.6	2.4
Text Sum.	5.1	6.9	7.0	**4.7**	6.8	8.3	21.0	13.4	10.9	**21.8**	15.1	1.7
Machine Tr.	**6.2**	6.4	9.9	7.5	8.3	10.0	16.8	12.0	**19.2**	9.6	9.6	4.0
Overall	**10.4**	12.7	15.1	14.5	16.5	17.2	**21.6**	15.1	15.9	20.1	15.9	10.0

Table 2: Task-level results on the benchmark. Left: PredSet mean absolute error (MAE; lower is better). Right: QnASet accuracy (higher is better). LRA = Litmus (Re)Agent, L++ = LITMUS++, TA = ThoughtAgent, SA = Single Agent, GPT = GPT-4.1, and M1 = Magentic-One.

improvements introduced in this work account for the gain over LITMUS++ (2.4 MAE reduction). Notably, the Single Agent (14.5) outperforms the ThoughtAgent (15.1) on MAE, suggesting that the multi-agent group chat overhead does not always help without DAG-level decomposition.

5.2 Per-Task Analysis

Table 2 present the per-task results. On PREDSET, LITMUS (RE)AGENT achieves the lowest MAE on four of six tasks, with the largest gain in code generation (14.2 vs. 39.8 for GPT-4.1), where evidence decomposition and feature-informed analysis are particularly effective. Both the ThoughtAgent (18.9) and Single Agent (17.8) substantially outperform GPT-4.1 on this task, indicating that tool use alone provides benefits even without DAG structure. Text summarisation is relatively easy for all systems (MAE 4.1–7.8), with the Single Agent achieving the best result (4.1). Mathematical reasoning remains challenging, and is the only task where GPT-4.1 performs best (18.2), likely reflecting weaker dependence on external evidence. Magentic-One consistently performs poorly across tasks, suggesting that general-purpose orchestration does not transfer well without task-specific design.

On QNASET, LITMUS (RE)AGENT achieves the highest overall accuracy (21.6%) and leads on code generation (55.3%) and classification/NLI (16.0%). Other systems achieve the best performance on individual tasks (e.g., ThoughtAgent on machine translation, LITMUS++ on QA/VQA, Single Agent on summarisation), but overall accuracy remains low across tasks. This indicates that comparative multilingual reasoning under incomplete evidence is substantially harder than numeric prediction. A full task–scenario breakdown is provided in Appendix Table 6.

5.3 Per-Scenario Analysis

Table 3 show results across the five evidence scenarios. LITMUS (RE)AGENT achieves the lowest MAE in all scenarios. The largest improvement appears in S1, where direct evidence must be correctly retrieved and synthesised (9.4 vs. 17.9 for ThoughtAgent). In transfer-heavy settings, performance differences widen: in S4, LITMUS (RE)AGENT maintains a low MAE (11.7) while Magentic-One degrades sharply (23.5), and the ThoughtAgent and Single Agent perform similarly (16.3 vs. 15.0), indicating limited benefit from multi-agent group chat without DAG decomposition.

On QNASET, LITMUS (RE)AGENT achieves the highest overall accuracy (21.6%) and leads in four of five scenarios. Gains are most pronounced in S3 (33.3%), suggesting that hypothesis decomposition is particularly effective for transfer from related languages. In S2, the Single Agent achieves the highest accuracy (22.0%), while the ThoughtAgent lags (9.3%), indicating that group chat overhead can be counterproductive for simpler cross-model transfer. Magentic-One shows a sharp drop in S4 and S5 (2.1% and 3.6%), highlighting the importance of domain-specific design in low-evidence settings. As in the per-task results, PREDSET MAE is lowest in S3 rather than S1, indicating that the scenarios do not form a strict difficulty ordering. Instead, they represent different evidence configurations, where close-language transfer can provide cleaner signals than noisy direct evidence.

5.4 Agent Reasoning Quality

We analyse the internal behaviour of LITMUS (RE)AGENT using five diagnostic metrics over 1,730 conversations (Table 4): hypothesis alignment with expert guidance (*thought faithfulness*), adherence to tool capabilities (*capability compliance*), relevance of retrieved documents (*web search relevance*), validity of selected features (*fea-*

	MAE (↓)						Accuracy % (↑)					
Scenario	**LRA**	**L++**	**TA**	**SA**	**GPT**	**M1**	**LRA**	**L++**	**TA**	**SA**	**GPT**	**M1**
S1: Same L+M	**9.4**	13.5	17.9	14.6	14.5	15.1	**29.3**	17.3	26.7	22.7	12.0	22.7
S2: Same L, Diff M	**10.3**	12.8	15.7	15.7	16.0	16.9	14.7	12.0	9.3	**22.0**	12.7	6.7
S3: Similar L+M	**9.0**	10.3	11.2	12.5	15.4	14.7	**33.3**	21.1	19.7	27.9	29.9	14.3
S4: Distant L+M	**11.7**	12.2	16.3	15.0	19.1	23.5	**13.8**	13.1	11.7	13.8	11.0	2.1
S5: Diff L+M	**11.9**	15.7	14.7	15.6	17.7	16.7	**16.6**	11.5	11.5	13.7	13.7	3.6
Overall	**10.4**	12.7	15.1	14.5	16.5	17.2	**21.6**	15.1	15.9	20.1	15.9	10.0

Table 3: PredSet MAE and QnASet accuracy by scenario. Abbreviations as in Table 2. Best per row in **bold**.

Metric	Value
Thought faithfulness	93.9%
Capability compliance	70.0%
Web search relevance	83.5%
Feature correctness	91.6%
Code execution success	30.6%

Table 4: Agent reasoning quality metrics for LITMUS (RE)AGENT, computed over 1,730 conversations and 5,184 generated hypotheses.

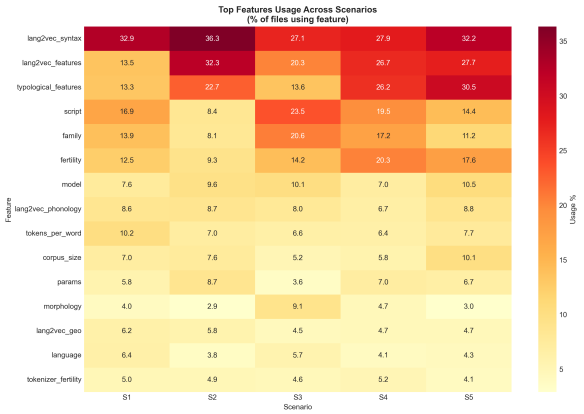


Figure 4: Feature usage heatmap across tasks. Rows represent feature categories (typological, preprocessing, model-based); columns represent tasks. Darker cells indicate higher usage.

ture correctness), and successful code execution. The system performs strongly on thought faithfulness, web search relevance, and feature correctness, but more weakly on code execution (30.6%). This suggests that hypothesis generation and evidence grounding are generally reliable, while program execution remains a bottleneck. Allowing retries raises execution success to 54.6%, but substantial room for improvement remains.

Appendix Figure 15 shows that the Coder agent primarily uses Random Forest, Ridge Regression, and XGBoost across tasks, while Figure 4 highlights extensive use of typological and phylogenetic features from `lang2vec` alongside preprocessing and model-based signals. Additional breakdowns and scenario-wise code quality metrics are reported in Appendix H.

We further assess output quality using an LLM-judge framework that scores each system's responses on four dimensions (1–5 scale): predictive plausibility, feature selection, coherence, and citation emphasis. Figure 5 shows the overall averages across all tasks. LITMUS (RE)AGENT achieves the highest scores on predictive plausibility (4.6) and feature selection (4.8), reflecting effective hypothesis formation and linguistic feature usage. GPT-4.1 scores highest on coherence (5.0) and citation emphasis (4.3), likely because its single-pass output is more consistently formatted. The ThoughtAgent closely tracks LITMUS (RE)AGENT on plausibility and feature selection, while LITMUS++ shows markedly lower scores across all dimensions, confirming the impact of the three proposed improvements. A detailed per-task breakdown is provided in Appendix Table 3.

5.5 Ablation Study: Backbone LLM

We evaluate the impact of the backbone LLM to assess whether gains arise from the DAG-based architecture or the underlying model. We replace GPT-4.1 with DeepSeek-V3, Llama-3.3-70B-Instruct, and o4-mini, and evaluate on code generation across S1, S3, and S5.

Table 5 reports LLM-judge metrics (1–5 scale) on shared queries. GPT-4.1 achieves the best overall performance, particularly on predictive plausibility and feature selection. o4-mini is closest, matching coherence but trailing on citation emphasis. DeepSeek-V3 performs well in S3 but drops in S1 and S5, while Llama-3.3-70B performs worst, especially on citation grounding. Overall, stronger backbones improve reasoning quality, but consistent trends across models indicate that gains primarily arise from the structured, evidence-grounded workflow rather than the backbone alone.

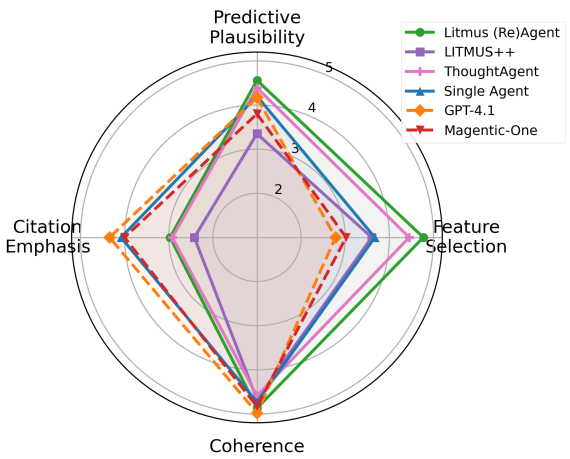


Figure 5: LLM-judge quality metrics (1–5) averaged across all tasks for each system. LITMUS (RE)AGENT leads on predictive plausibility and feature selection; GPT-4.1 leads on coherence and citation emphasis.

Backbone	Scen.	Pred.	Feat.	Coh.	Cite.
GPT-4.1	S1	**4.8**	**4.9**	4.9	**3.2**
	S3	**4.7**	**4.7**	4.9	**2.9**
	S5	4.5	**4.9**	**5.0**	**3.0**
o4-mini	S1	4.0	4.4	**5.0**	2.8
	S3	4.2	**4.7**	4.8	2.7
	S5	**4.4**	4.8	**5.0**	2.6
DeepSeek-V3	S1	3.6	3.7	4.6	2.0
	S3	4.3	4.3	4.9	2.3
	S5	4.0	3.8	4.9	2.4
Llama-3.3-70B	S1	3.4	3.8	4.2	1.1
	S3	3.6	3.3	4.1	1.1
	S5	3.9	3.6	4.4	1.3

Table 5: Backbone LLM ablation on code generation (S1, S3, S5). Quality metrics from LLM-judge evaluation (1–5 scale) computed on common questions across all models. Pred. = Predictive Plausibility, Feat. = Feature Selection, Coh. = Coherence, Cite. = Citation Emphasis.

5.6 Human Evaluation Study

We conduct a within-subjects study with 8 participants (4 novice, 4 expert), each answering 10 code-generation prediction questions: 5 using general LLM tools (ChatGPT, Gemini, Claude) and 5 using LITMUS (RE)AGENT, under the same constrained paper corpus. Experts have 1–5+ years of NLP experience; novices are undergraduate students. For each question, participants report predictions, reasoning, confidence, and rate five Likert dimensions: confidence (0–5), and interpretability, explainability, justification, and actionability (1–5). Time per question is recorded.

Table 8 summarises results. Participants consistently rate predictions higher with LITMUS (RE)AGENT, with the largest gains in justification (+1.0) and actionability (+0.9), reflecting improved reasoning and evidence grounding. Novices benefit more overall (e.g., confidence +1.0, actionability +1.0), while experts show the largest gain in justification (+1.1). Time increases for novices (13.7 vs. 5.9 min) but remains stable for experts (10.3 vs. 10.7 min). In a final survey, 5 of 7 participants report improved prediction accuracy. Figure 6 shows quality profiles by participant type; full results are in Appendix F.

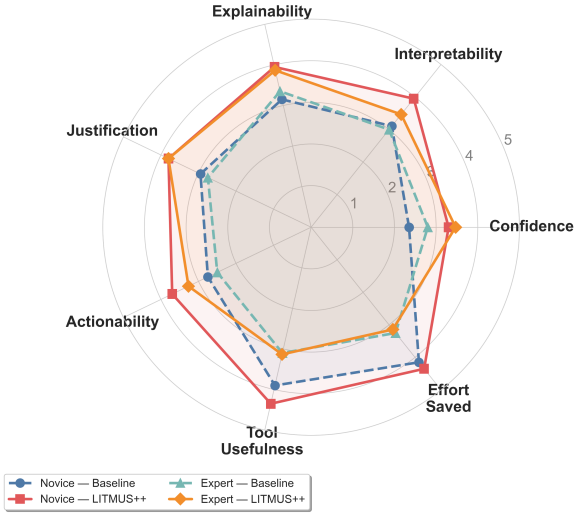


Figure 6: Human evaluation quality metrics by participant type (novice vs. expert) and phase (baseline vs. LITMUS (RE)AGENT).

6 Conclusion

We presented LITMUS (RE)AGENT and a controlled benchmark for multilingual performance prediction under incomplete evidence, spanning six tasks and diverse evidence scenarios. LITMUS (RE)AGENT combines DAG-based agent orchestration with citation-grounded retrieval, linguistic feature modelling, and regression-based validation to enable structured predictive analysis. Our results show that structured, hypothesis-driven workflows consistently improve performance, particularly in transfer-heavy settings where direct evidence is limited. We further find that multi-agent group chat alone is insufficient without DAG-level decomposition, highlighting the importance of explicit reasoning structure. Overall, this work demonstrates that agentic systems can support reliable and transparent multilingual performance estimation under realistic evidence constraints, and provides a reusable benchmark for future research in predictive evaluation.

7 Limitations

This work has three main limitations. **System:** all agentic experiments use GPT-4.1, so generalisation to smaller or open-source backbones is unverified; Coder-agent execution success is moderate (30.6% at the conversation level), and end-to-end latency remains higher than that of direct prompting. **Benchmark:** coverage spans six tasks but not multimodal reasoning, dialogue, or safety; predictions depend on a fixed, curated corpus; and paper-reported ground truth may include measurement noise and cross-study inconsistency. **Comparison:** while the six-system evaluation provides a broad comparison, LITMUS++, Magentic-One, and the ThoughtAgent use different internal coordination strategies, so gains over them reflect both architectural and prompt-level differences that are not fully disentangled.

Ethical Considerations

LITMUS (RE)AGENT is intended to support multilingual evaluation planning, not to replace direct benchmarking when high-stakes decisions require ground-truth measurement. Predictive estimates can be useful for prioritisation and early-stage analysis, but they may inherit biases from the published literature, including uneven language coverage, inconsistent reporting standards, and systematic underrepresentation of low-resource settings. To mitigate these risks, we evaluate under controlled evidence restriction, retain explicit citations to accessible evidence, and emphasise that predictions should be treated as decision-support signals rather than authoritative substitutes for benchmark results.

References

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6430–6439.

Kabir Ahuja, Sandipan Dandapat, Sunayana Sitaram, and Monojit Choudhury. 2022a. Beyond static models and test sets: Benchmarking the potential of pre-trained models across tasks and languages. In *Proceedings of NLP Power! The First Workshop on Efficient Benchmarking in NLP*, pages 64–74.

Kabir Ahuja, Shanu Kumar, Sandipan Dandapat, and Monojit Choudhury. 2022b. Multi task learning for zero shot performance prediction of multilingual models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5454–5467.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Błażej Dolicki and Gerasimos Spanakis. 2021. Analysing the impact of linguistic features on cross-lingual transfer. *arXiv preprint arXiv:2105.05975*.

Adam Fourney, Gyu Hong, Lance J.Y. Liew, Jianping Chi, Ziyang Wang, Ryen White, Eric Horvitz, Mounia Lalmas, Faraaz Ansari, Chris Ackerson, Ben Suh, Javier Diaz, Yoni Halpern, Mark Dredze, Jacob Eisenstein, Monica S. Lam, and Nataniel Ruiz. 2024. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2410.04468*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International conference on machine learning*, pages 4411–4421. PMLR.

Shanu Kumar, Soujanya Abbaraju, Sandipan Dandapat, Sunayana Sitaram, and Monojit Choudhury. 2023. Ditto: A feature representation imitation approach for improving cross-lingual transfer. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 385–406.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K"uttler, Mike Lewis, Wen-tau Yih, Tim Rockt"aschel, and Sebastian Riedel. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Zihao Li, Yucheng Shi, Zirui Liu, Fan Yang, Ali Payani, Ninghao Liu, and Mengnan Du. 2025. Language ranker: A metric for quantifying llm performance across high and low-resource languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28186–28194.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, and 1 others. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and 1 others. 2020. Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, and 1 others. 2019. Choosing transfer languages for cross-lingual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3125–3135.

Patrick Littell, David R Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14.

Qian Liu, Xiaoting Wang, Yining Hou, Bohan Lyu, Huanliang Liu, Yunhao Liu, Haowen Jiang, Chaojia Gu, Hanqing Lu, Xintong Han, Weiliang Dai, Cailong Cai, Yizhou Wang, Zhiyuan Liu, Lei Hou, and Juanzi Li. 2023. Paperqa: Citation-aware question answering over scientific literature. *arXiv preprint arXiv:2310.12321*.

Yixin Liu, Guibin Zhang, Kun Wang, Shiyuan Li, and Shirui Pan. 2025. Graph-augmented large language model agents: Current progress and future prospects. *arXiv preprint arXiv:2507.21407*.

Avni Mittal, Shanu Kumar, Sandipan Dandapat, and Monojit Choudhury. 2025. Litmus++: An agentic system for predictive analysis of low-resource languages across tasks and models. In *Proceedings of The 14th International Joint Conference on Natural Language Processing and The 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 47–54.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 1797–1807.

Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. 2020. Leep: A new measure to evaluate transferability of learned representations. In *International conference on machine learning*, pages 7294–7305. PMLR.

NLLB Team. 2022. No language left behind: Scaling human-centered machine translation. https://arxiv.org/abs/2207.04672.

OpenAI. 2023. Gpt-4 technical report. https://cdn.openai.com/papers/gpt-4.pdf.

Sebastian Ruder, Noah Constant, Jan Botha, Aditya Siddhant, Orhan Firat, Jinlan Fu, Pengfei Liu, Junjie Hu, Dan Garrette, Graham Neubig, and 1 others. 2021. Xtreme-r: Towards more challenging and nuanced multilingual evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10215–10245.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Viktoria Schram, Daniel Beck, and Trevor Cohn. 2023. Performance prediction via bayesian matrix factorisation for multilingual natural language processing tasks. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1790–1801.

Anirudh Srinivasan, Gauri Kholkar, Rahul Kejriwal, Tanuja Ganu, Sandipan Dandapat, Sunayana Sitaram, Balakrishnan Santhanam, Somak Aditya, Kalika Bali, and Monojit Choudhury. 2022. Litmus predictor: An ai assistant for building reliable, high-performing and fair multilingual nlp systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 13227–13229.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*.

Soumava Tan, Lipika Dey, Kushagra Chawla, Sumeet Kumar, Mihir Rajora, Rishabh Mehrotra, Ahmed Awadallah, and Mitesh M. Khapra. 2024. Judgebench: A benchmark for evaluating llm-based judges. https://openreview.net/forum?id=G0dksFayVq.

Alexander Tsvetkov and Alon Kipnis. 2024. Information parity: Measuring and predicting the multilingual capabilities of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7971–7989.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. Logme: Practical assessment of pre-trained models for transfer learning. In *International conference on machine learning*, pages 12133–12143. PMLR.

Tianhao Zhang, Hui Lu, Yilin Zhang, Saibin Zhang, Yuhang Jiang, Huanxin Wang, and Peiyan Liu. 2024a. Deepresearcher: Autonomous agents for end-to-end literature review and reasoning. *arXiv preprint arXiv:2402.06789*.

Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2024b. On the diagram of thought. *arXiv preprint arXiv:2409.10038*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.

A Agent Lifecycle Details

Each *ThoughtAgent* transitions between three core states: **Active**: investigating a hypothesis with assigned tools; **Completed**: finished investigation and returned validated evidence; **Discarded**: pruned when deemed irrelevant, redundant, or divergent. State transitions are managed by the *ThoughtAnalyzerAgent*, which monitors progress and determines whether to continue, complete, or discard a branch. This lifecycle ensures that only relevant outputs contribute to the final analysis while preserving an auditable trace of reasoning paths.

B Systems Compared

We compare the reported systems under the same reduced-corpus evidence restriction. All agentic systems use GPT-4.1 as the underlying LLM.

LITMUS (RE)AGENT (ours). Our full system answers each query using a dynamic DAG of specialised Thought Agents together with the three improvements introduced in Section 4.2: enriched expert knowledge, enhanced code execution with linguistic feature libraries, and prompt stabilisation. Queries are decomposed into hypothesis-specific branches that are investigated in parallel and then aggregated.

LITMUS++ (prior DAG system). The predecessor system (Mittal et al., 2025) uses the same DAG-based architecture with multiple Thought Agents, but without the three improvements introduced in this work: the knowledge base is smaller, the Coder agent lacks access to `lang2vec` and URIEL features, and prompts follow the original design. Comparing LITMUS++ to LITMUS (RE)AGENT isolates the effect of the three proposed changes.

ThoughtAgent (single multi-agent group chat, no DAG). A single ThoughtAgent that operates as a multi-agent group chat of specialised sub-agents—Research Planner, Web Search and Crawl, Expert Knowledge, Coder, and Reporter—but without DAG-level decomposition into parallel hypotheses. It processes the full query in one thread through coordinated sub-agent interaction. This baseline isolates the contribution of the multi-agent group chat architecture when hypothesis parallelism is absent.

Single Agent (ReAct loop, no group chat). A single ReAct agent with direct access to code execution, web search, and expert knowledge base tools in one agent loop, without the multi-agent group chat or DAG decomposition. This is the simplest agentic baseline and isolates the value of tool-augmented reasoning without any multi-agent coordination.

GPT-4.1 Direct Baseline. This baseline answers each query in a single GPT-4.1 call using the available evidence provided in-context, without iterative tool use or agentic decomposition. It measures the value added by agentic scaffolding beyond direct prompting with access to the same restricted evidence.

Magentic-One (Fourney et al., 2024). A general-purpose multi-agent framework that coordinates multiple agents through a lead orchestrator agent. Unlike LITMUS (RE)AGENT, it is not designed specifically for multilingual evaluation and does not include domain-specific tools, linguistic feature libraries, or a curated knowledge base. Including Magentic-One tests whether a general-purpose multi-agent approach can match domain-specific design on this task.

B.1 LLM Hyperparameters

All systems use GPT-4.1 via the Azure OpenAI API (version `2025-01-01-preview`). The main agentic systems (LITMUS (RE)AGENT, LITMUS++, ThoughtAgent, and Single Agent) use a default configuration with temperature 1.0 and top-p 1.0. The GPT-4.1 direct baseline uses temperature 0.7 with a maximum output length of 2,000 tokens.

The Expert Knowledge (Expert KB) agent operates with temperature 0 for deterministic retrieval and is augmented with a web search database. Retrieved documents are filtered using cosine similarity with a threshold of 0.90, and the top-k results ($k = 2$) are selected. A caching mechanism is used across experiments for efficiency. For structured output parsing, a separate LLM call with higher token limits and fallback handling is used. All evaluation is performed using GPT-4.1 as the LLM judge.

C Implementation Details

Tooling. The backend orchestration uses *Autogen* for multi-agent coordination, *ChromaDB* as the vector store for the curated knowledge base, and the Firecrawl API for web search and scraping. ThoughtAgents invoke modular tools for search, scraping, PDF/JSON parsing, knowledge-base retrieval, and lightweight regression.

Deployment. The system runs locally or via a hosted interface, including a Dockerised setup for reproducibility. External search requires a user-provided API key, while other components operate over the benchmark corpus.

Outputs. Final predictions are returned together with supporting citations and structured reasoning traces. When applicable, the system also returns an uncertainty estimate produced during the aggregation stage.

Extensibility. The DAG orchestration is model- and task-agnostic: adding new tools or agents requires registration in the orchestration graph without changing the overall control flow. Related details on agent lifecycle and knowledge-base curation are provided in Appendices A and D.

D Knowledge Base Curation

A curated multilingual knowledge base supports LITMUS (RE)AGENT during hypothesis formation and feature-informed analysis. It integrates (i) **literature-derived resources** from peer-reviewed papers, benchmarks, and typological databases (Lauscher et al., 2020; Dolicki and Spanakis, 2021; Srinivasan et al., 2022; Ahuja et al., 2022a; Kumar et al., 2023), and (ii) structured expert-oriented notes used to guide analysis strategies in sparse-evidence settings. The knowledge base is organised around language–task–model relations together with reusable analytical patterns, such as common transfer assumptions, feature types, and failure modes encountered in multilingual evaluation. In the current version, the knowledge base is used as structured guidance rather than as ground truth itself.

E Detailed Task–Scenario Results

Table 6 reports PredSet MAE, QnASet accuracy, and LLM-judge quality scores for each task across all six systems.

E.1 Per-Metric MAE Analysis

Figure 7 shows the mean absolute error of LITMUS (RE)AGENT broken down by evaluation metric type. Accuracy-based metrics (used primarily in code generation, classification/NLI, and mathematical reasoning) exhibit the highest MAE (12.7), reflecting the wider score ranges and greater prediction difficulty in these tasks. In contrast, text-overlap metrics like ROUGE (6.5), BLEU (5.0), and chrF (7.5), show substantially lower MAE. This pattern is consistent with the per-task results in Table 2: tasks evaluated with accuracy metrics tend to produce more variable scores, while summarisation and translation metrics operate on narrower, more predictable ranges. These results suggest that metric-aware normalisation or task-specific calibration may further improve prediction quality.

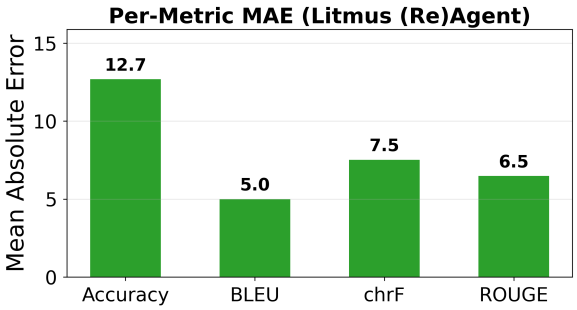


Figure 7: Per-metric MAE for LITMUS (RE)AGENT. Accuracy-based metrics show higher prediction error than text-overlap metrics such as ROUGE, BLEU, and chrF.

E.2 Per-Scenario MAE Trends

Figure 8 presents the per-scenario MAE as a line chart, showing how each system's prediction error evolves across the five evidence configurations. Several patterns are visible. First, LITMUS (RE)AGENT maintains the lowest MAE across all scenarios, with a relatively flat trajectory (range 9.0–11.9). Second, all systems show increased

		PredSet MAE (↓)						QnASet Accuracy % (↑)						Quality (1–5) (↑)				
Task	**Sys**	**S1**	**S2**	**S3**	**S4**	**S5**	**Avg**	**S1**	**S2**	**S3**	**S4**	**S5**	**Avg**	**Pr**	**Ft**	**Co**	**Ci**	**Avg**
Code Gen.	LRA	**7.9**	**14.4**	**8.5**	20.2	20.4	**14.2**	**88.0**	32.0	**73.9**	38.1	**40.0**	**55.3**	**4.7**	**4.9**	4.9	3.0	**4.4**
	L++	10.5	29.9	22.7	**18.4**	24.6	21.1	32.0	16.0	34.8	19.0	25.0	25.4	3.1	3.3	4.6	2.1	3.3
	TA	16.8	23.6	12.0	24.7	18.6	18.9	60.0	20.0	43.5	42.9	30.0	39.5	4.6	4.7	4.8	3.0	4.3
	SA	12.0	23.5	12.0	21.8	21.1	17.8	76.0	**48.0**	65.2	**47.6**	15.0	51.8	4.4	3.9	4.9	4.2	4.4
	GPT	28.4	43.7	39.5	47.7	39.8	39.8	24.0	20.0	43.5	23.8	25.0	27.2	4.4	3.3	**5.0**	**4.6**	4.3
	M1	17.8	30.9	22.4	38.7	**26.0**	27.0	84.0	24.0	69.6	9.5	10.0	41.2	3.8	3.0	4.9	4.2	4.0
Math Reas.	LRA	**14.0**	17.6	26.2	26.6	14.8	20.0	12.0	**24.0**	25.0	**12.5**	4.0	15.5	**4.6**	**4.9**	4.9	3.0	**4.3**
	L++	28.0	23.7	31.7	26.7	32.5	28.8	**16.0**	8.0	12.5	**12.5**	4.0	10.6	3.2	3.5	4.7	2.2	3.4
	TA	25.5	25.0	27.2	41.4	19.6	27.5	**32.0**	8.0	16.7	8.3	0.0	13.0	4.5	4.6	4.7	2.8	4.2
	SA	30.0	22.8	29.0	29.8	36.4	29.6	4.0	12.0	16.7	4.2	**12.0**	9.8	4.3	3.9	4.9	4.2	4.3
	GPT	18.0	**6.7**	**23.3**	**20.8**	**12.3**	**18.2**	12.0	**24.0**	**29.2**	**16.7**	**12.0**	**18.7**	4.1	2.8	**5.0**	**4.0**	4.0
	M1	23.0	15.4	28.5	37.9	22.1	26.8	12.0	4.0	12.5	4.2	4.0	7.3	3.7	2.8	4.8	3.9	3.8
QA/VQA	LRA	13.4	11.7	7.2	**5.7**	8.7	9.5	0.0	0.0	28.0	4.0	8.0	8.0	**4.5**	**4.8**	4.8	2.9	**4.2**
	L++	14.9	**6.7**	**3.6**	**4.7**	14.7	**8.7**	**20.0**	**16.0**	**24.0**	**32.0**	**20.0**	**22.4**	4.2	4.6	4.9	2.7	4.1
	TA	23.3	14.6	12.3	10.9	10.9	14.7	20.0	4.0	12.0	4.0	0.0	8.0	4.1	4.2	4.3	2.7	3.8
	SA	**12.0**	18.2	11.6	9.0	11.6	12.4	**24.0**	**20.0**	8.0	12.0	20.0	16.8	4.2	3.8	4.7	4.0	4.2
	GPT	13.0	8.8	8.4	6.9	11.1	9.6	**20.0**	**16.0**	**24.0**	4.0	16.0	16.0	4.1	2.7	**5.0**	**4.1**	4.0
	M1	17.8	13.7	12.0	12.3	**13.7**	14.1	**20.0**	4.0	4.0	0.0	0.0	5.6	3.7	3.1	4.7	3.6	3.8
Class./NLI	LRA	**8.7**	8.6	**3.2**	**4.8**	13.5	**7.9**	**36.0**	8.0	**16.0**	**12.0**	**8.0**	**16.0**	**4.6**	**4.7**	4.9	3.0	**4.3**
	L++	14.1	**8.8**	5.5	12.3	**8.3**	9.6	8.0	**16.0**	0.0	**12.0**	0.0	7.2	3.1	3.4	4.6	2.4	3.4
	TA	15.0	7.9	5.5	7.9	17.7	11.6	8.0	4.0	12.0	4.0	4.0	6.4	4.2	4.2	4.3	2.8	3.9
	SA	14.7	14.9	5.8	10.3	10.4	11.9	8.0	**20.0**	**20.0**	**16.0**	4.0	13.6	4.3	3.6	4.7	4.2	4.2
	GPT	12.2	**8.8**	3.4	10.9	13.5	9.9	8.0	8.0	**20.0**	**12.0**	0.0	9.6	4.3	2.5	**5.0**	**4.5**	4.1
	M1	14.8	14.9	5.7	12.5	11.0	12.0	8.0	0.0	4.0	0.0	0.0	2.4	3.9	3.2	4.8	3.9	4.0
Text Sum.	LRA	2.5	6.1	2.7	8.4	14.2	**6.5**	8.0	16.0	36.0	16.0	31.6	21.0	**4.5**	**4.6**	4.9	2.9	**4.2**
	L++	4.7	18.1	3.1	11.3	23.3	12.6	4.0	8.0	**44.0**	0.0	10.5	13.4	3.3	3.4	4.6	2.5	3.4
	TA	4.7	14.8	**2.0**	12.4	15.8	9.8	0.0	12.0	20.0	8.0	15.8	10.9	4.4	4.4	4.6	2.9	4.1
	SA	**2.2**	**2.3**	3.8	**6.7**	20.1	7.1	**12.0**	**28.0**	**44.0**	8.0	15.8	**21.8**	4.0	3.5	4.5	3.8	4.0
	GPT	2.9	14.0	**1.9**	15.6	17.0	9.6	4.0	8.0	40.0	4.0	21.1	15.1	4.2	2.8	**5.0**	**4.6**	4.1
	M1	3.7	12.2	2.8	15.9	**14.2**	9.2	4.0	4.0	0.0	0.0	0.0	1.7	3.8	3.0	4.8	4.0	3.9
Machine Tr.	LRA	9.4	5.6	8.5	**5.1**	**4.1**	**6.2**	**32.0**	8.0	**24.0**	4.0	**16.0**	16.8	**4.7**	**4.8**	4.9	3.1	**4.4**
	L++	**4.4**	6.9	**5.8**	7.2	6.6	6.4	24.0	8.0	12.0	4.0	12.0	12.0	3.2	3.4	4.6	2.6	3.5
	TA	14.2	10.2	10.6	8.2	7.9	9.9	**40.0**	8.0	16.0	**8.0**	**24.0**	**19.2**	4.6	4.6	4.8	3.2	4.3
	SA	8.2	7.3	9.2	6.9	6.7	7.5	12.0	4.0	16.0	0.0	16.0	9.6	4.0	3.4	4.6	4.0	4.0
	GPT	9.5	8.9	9.3	7.3	7.4	8.3	4.0	0.0	**24.0**	**8.0**	12.0	9.6	4.1	2.7	**5.0**	**4.3**	4.0
	M1	9.8	8.8	8.0	13.7	9.1	10.0	8.0	4.0	0.0	0.0	8.0	4.0	3.9	3.1	4.9	4.5	4.1

Table 6: Detailed results by task: PredSet MAE (↓), QnASet accuracy % (↑), and LLM-judge quality (1–5, ↑). Quality: Pr = Predictive Plausibility, Ft = Feature Selection, Co = Coherence, Ci = Citation Emphasis. Abbreviations as in Table 2.

MAE in S4 (distant language transfer), but this effect is most pronounced for Magentic-One (23.5), which lacks the linguistic feature tools needed for cross-lingual reasoning. Third, S3 (similar language transfer) often yields lower MAE than S1 (direct evidence), confirming the non-monotonic difficulty pattern discussed in Section 5. Fourth, the gap between LITMUS (RE)AGENT and the other systems widens in S4, indicating that the system's typological feature extraction and regression capabilities provide the largest benefit under distant transfer.

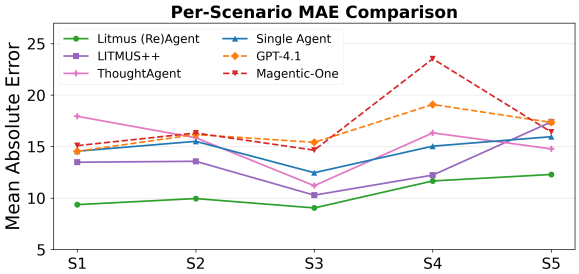


Figure 8: Per-scenario MAE trends across all five systems. LITMUS (RE)AGENT maintains the lowest and most stable MAE, while Magentic-One degrades sharply in S4 (distant language transfer).

E.3 Detailed Error Analysis by Task and Scenario

Figures 9 and 10 provide a finer-grained analysis of LITMUS (RE)AGENT's prediction errors across task–scenario combinations.

Figure 9 shows the mean MAE broken down by scenario (x-axis) with each task as a separate bar. Code generation and mathematical reasoning exhibit notably higher MAE than other tasks across most scenarios, consistent with the per-task results in Table 2. Within code generation, S2 and S5 show elevated error, suggesting that cross-model and combined cross-lingual/cross-model transfer are particularly challenging for this task. In contrast, text summarisation and machine translation maintain consistently low MAE across all scenarios, indicating that the system's predictions are more stable for tasks evaluated with text-overlap metrics.

Figure 10 shows the full distribution of per-question absolute errors using box plots on a log scale. The distributions reveal that median errors are often low even when mean MAE is high, indicating that a small number of high-error predictions drive up the mean. This is most visible in code generation and mathematical reasoning under S4 and S5, where the interquartile range is wide and outliers are frequent. QA/VQA, classification/NLI, and machine translation show tighter distributions with fewer extreme errors. These patterns suggest that targeted improvements in high-error cases—particularly for code generation in distant-transfer scenarios—could yield substantial gains in overall MAE.

F Human Evaluation Details

This appendix provides additional analysis from the human evaluation study described in Section 5.6.

F.1 Participant Demographics

The 8 participants comprise two groups. The **novice** group (4 participants) consists of undergraduate students in computer science or AI/ML programmes with less than one year of NLP evaluation experience and no prior exposure to multilingual benchmarking. The **expert** group (4 participants) includes a faculty member with a PhD in AI/NLP and 5+ years of experience, two researchers with 1–3 years of experience in code generation and speech/AI agents respectively, and a PhD student in NLP with 3–5 years of experience. Participants were recruited through direct invitation. Participation was voluntary and no monetary compensation was provided. All participants completed the baseline phase (5 questions each); most completed the LITMUS (RE)AGENT phase, though two experts answered fewer than 5 questions due to time constraints.

F.2 Study Interface and Evaluation Form

The human evaluation was conducted using LITMUS (RE)AGENT, an interactive tool built on top of the LITMUS++ benchmark. Figures 11 and 12 show representative screenshots of the tool interface. The system presents each prediction question along with a constrained set of accessible papers, and provides structured outputs including retrieved evidence, linguistic feature analysis, and regression-based predictions. Participants interact with the system in a conversational format and can request additional analysis or clarification.

After each question, participants completed a structured evaluation form (Figure 13). The form captures the predicted value, confidence interval, reasoning strategy, challenges encountered, and Likert-scale ratings on five quality dimensions (confidence, interpretability, explainability, justification,

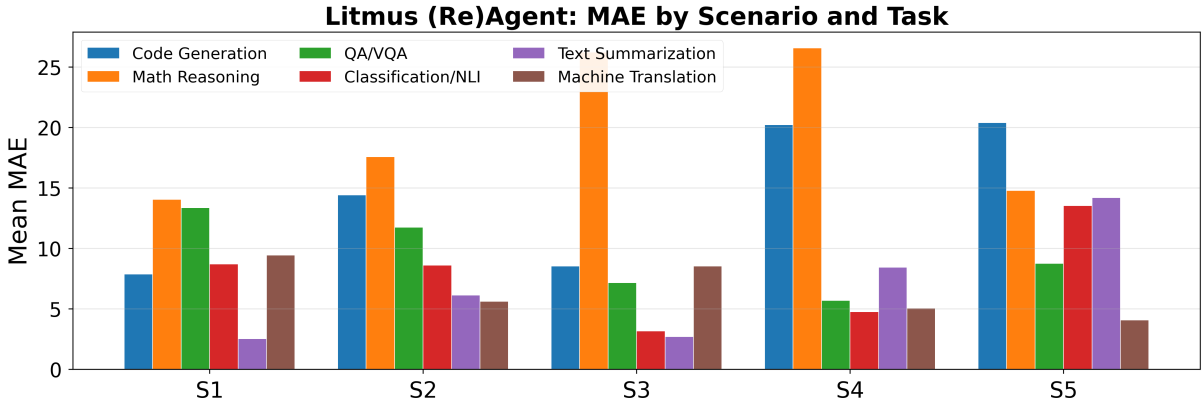


Figure 9: Litmus (Re)Agent: mean MAE by scenario and task. Each group of bars corresponds to one scenario (S1–S5), with tasks shown as different colours. Code generation and mathematical reasoning show higher and more variable error than other tasks.

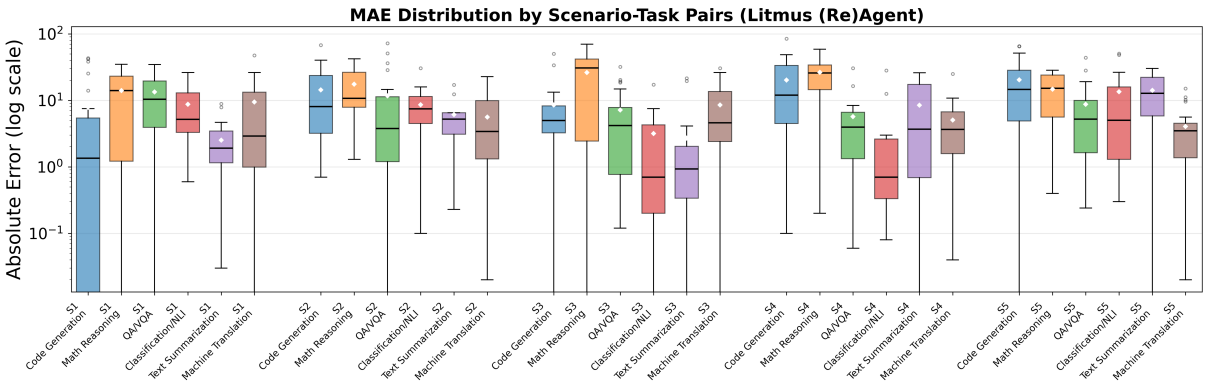


Figure 10: Distribution of per-question absolute errors for LITMUS (RE)AGENT, grouped by scenario–task pairs (log scale). Diamonds indicate means; horizontal lines indicate medians. Wider boxes and more outliers appear in code generation and math reasoning, especially in transfer-heavy scenarios.

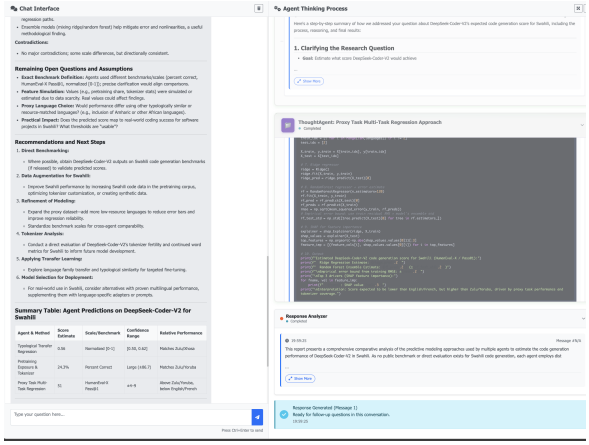


Figure 11: Sample screenshot of the LITMUS (RE)AGENT interface showing the conversational interaction, structured evidence retrieval, and prediction output presented to participants during the LITMUS (RE)AGENT phase.

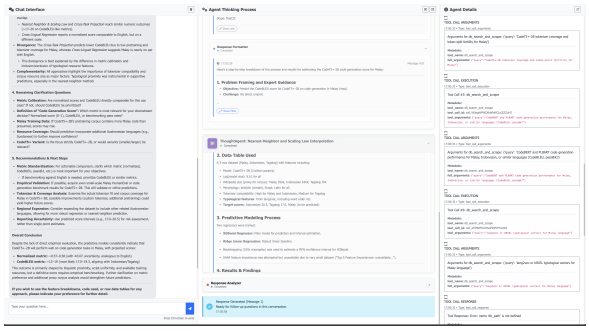


Figure 12: Sample screenshot of the LITMUS (RE)AGENT interface showing the final prediction summary with confidence intervals, reasoning trace, and supporting evidence.

and actionability). Tool usefulness and effort saved ratings are also collected. This form was identical across both the baseline and LITMUS (RE)AGENT phases to ensure comparability. The complete list of evaluation questions, detailed metric rubrics, and per-section form structure are provided in the subsections below.

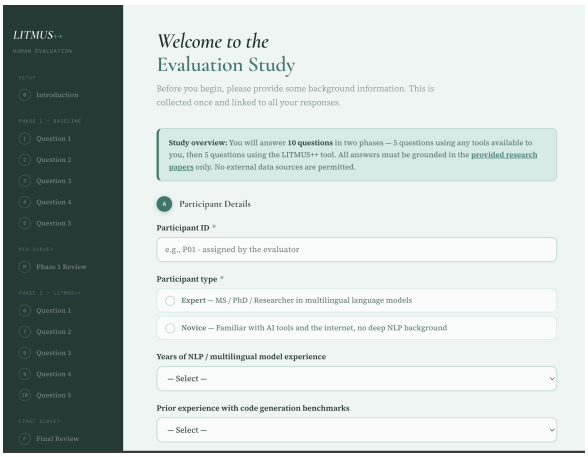


Figure 13: Screenshot of the human evaluation form used after each prediction question. Participants rate quality dimensions on Likert scales and report their reasoning strategies and challenges.

F.3 Evaluation Questions

Table 7 lists the 10 prediction questions used in the human evaluation study. All questions target code generation for low-resource or mid-resource languages, requiring participants to predict a numeric performance score (pass@1).

#	Phase	Model	Language
1	Baseline	GPT-4	Hindi
2	Baseline	GPT-4o	Odia
3	Baseline	GPT-3.5	Romanian
4	Baseline	DeepSeek-Coder-V2	Shona
5	Baseline	Claude-3.5-Opus	Bosnian
6	LITMUS++	CodeT5+-2B	Malay
7	LITMUS++	Claude-3.5-Opus	Macedonian
8	LITMUS++	GPT-4o	Bengali
9	LITMUS++	GPT-3.5	Dari
10	LITMUS++	DeepSeek-Coder-V2	Swahili

Table 7: The 10 code generation prediction questions used in the human evaluation.

F.4 Per-Question Form Structure

For each question, participants completed a structured form (Figure 13) with six sections:

Section A: Prediction. A numeric predicted value (required) and an optional 80% confidence interval (low–high range).

Section B: Reasoning Process. Strategy used (multi-select from: direct lookup, interpolation, cross-paper synthesis, analogical reasoning, tool output, domain knowledge, educated guess, other), step-by-step reasoning (free text), papers referenced, and number of papers searched.

Section C: Challenges. Multi-select from: missing data, inconsistent papers, too many papers, complex metric, tool limitation, unfamiliar language/model, no significant challenge.

Section D: Confidence and Interpretability. Four Likert-scale ratings with detailed rubrics:

- **Confidence** (0–5): 0 = no basis, random estimate; 1 = vague intuition; 2 = partially relevant info, significant uncertainty; 3 = relevant patterns, not directly reported; 4 = closely matching data, minor extrapolation; 5 = exact or near-exact data point found.
- **Interpretability** (1–5): 1 = gut feeling, cannot articulate steps; 3 = general approach clear, some steps fuzzy; 5 = every step explicit and traceable.
- **Explainability** (1–5): 1 = cannot give coherent explanation; 3 = broad strokes OK, cannot justify specifics; 5 = fully transparent with paper references.
- **Justification** (1–5): 1 = raw numbers, zero explanation; 3 = some factors explained, gaps remain; 5 = thorough, well-evidenced explanation.

Section E: Tool Experience. Tool usefulness (1–5: 1 = no relevant info; 3 = useful direction, verification needed; 5 = directly provided the answer), effort saved (1–5: 1 = no savings or wasted time; 5 = tool essentially answered it), better/worse prediction vs. unaided (5-point scale from "much worse" to "much better"), and optional free-text feedback.

Section F: Decision Quality. Actionability (1–5: 1 = too uncertain for any decision; 3 = one input among many; 5 = sufficient for deployment decisions) and calibration surprise (how surprised if prediction wrong by ± 5 points, from "not at all" to "extremely").

F.5 Mid-Survey and Final Survey

Between phases, participants completed a mid-survey assessing Phase 1 cognitive load (mental demand, physical effort, frustration, success feeling) and tool experience (overall utility, ease of use). After Phase 2, a final survey collected: overall cognitive load for both phases, phase comparison (which produced better predictions, ease comparison, confidence comparison), LITMUS++ evaluation (utility, ease, transparency, trust, future use, comparison rank vs. other tools), highlights and failures (easiest/hardest questions, specific tool failures), and open feedback (most valuable aspect, most limiting, feature request).

F.6 Quality Metric Profiles

Table 8 provides the full numeric breakdown of human evaluation results, including novice–expert subgroup analysis.

	Overall			Novice			Expert		
Metric	**B**	**L**	**Δ**	**B**	**L**	**Δ**	**B**	**L**	**Δ**
Confidence	2.6	3.4	**+.8**	2.4	3.3	**+1.0**	2.8	3.5	+.7
Interpretability	3.1	3.7	**+.7**	3.1	4.0	**+.8**	3.0	3.5	+.5
Explainability	3.2	3.9	**+.7**	3.2	4.0	**+.8**	3.4	3.9	+.5
Justification	2.8	3.8	**+1.0**	3.0	3.8	+.9	2.8	3.8	**+1.1**
Actionability	2.6	3.5	**+.9**	2.8	3.7	**+1.0**	2.5	3.3	+.8
Tool Useful.	3.5	3.8	+.3	3.9	4.4	+.5	3.1	3.1	.0
Effort Saved	3.7	3.9	+.2	4.2	4.4	+.2	3.3	3.1	−.1
Time (min)	8.6	12.0	+3.5	5.9	13.7	+7.7	10.7	10.3	−.3

Table 8: Human evaluation: baseline (B) vs. LITMUS (RE)AGENT (L). Higher is better except for time.

Figure 6 shows the radar chart of all seven quality metrics broken down by participant type and phase. Novices show larger gains across most dimensions when moving from baseline to LITMUS (RE)AGENT, particularly in interpretability and actionability. Experts start from a higher baseline on confidence and explainability but show a pronounced improvement in justification quality. Tool usefulness and effort saved ratings remain relatively flat for experts, suggesting they derive value from the system's reasoning traces rather than from reduced manual effort.

F.7 Self-Assessed Prediction Quality

Figure 14 shows quality dimensions and self-assessed prediction quality across the two phases. In the baseline phase, 60.0% of responses were rated "slightly better" or "much better" than what participants felt they could achieve unaided; in the LITMUS (RE)AGENT phase, this proportion rose to 65.7%. The distribution also shifted: "much better" responses increased from 15.0% to 22.9%, while "same" responses dropped from 22.5% to 11.4%.

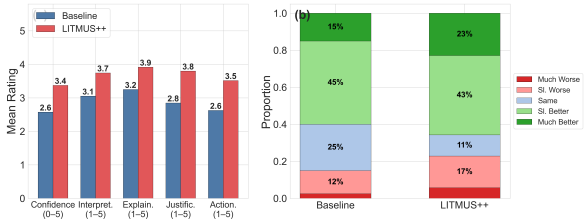


Figure 14: Human evaluation results: (a) mean quality dimension ratings (baseline vs. LITMUS (RE)AGENT) across all participants, (b) self-assessed prediction quality distribution showing the proportion of responses rated from "much worse" to "much better" relative to unaided prediction.

F.8 Strategy Shifts

In the baseline phase, participants relied on diverse strategies: interpolation (18), cross-paper search (14), tool output (14), analogical reasoning (11), and guessing (4). In the LITMUS++ phase, *tool-output* became the dominant strategy (23 uses), while guessing nearly disappeared (4→1). This shift reflects participants' reliance on the system's structured outputs rather than manual heuristics.

F.9 Challenge Distribution

The most common challenge across both phases was *missing-data* (31 baseline, 27 LITMUS++). LITMUS++ reduced the prevalence of *too-many* papers to process (10→5) and *tool-limitation* challenges (5→1), suggesting that the system's automated retrieval and synthesis pipeline alleviates manual search burden.

F.10 Calibration Confidence

Participants were asked how surprised they would be if the true answer differed from their prediction by ±5 percentage points. In the baseline phase, 25.0% reported "not at all" surprised, indicating low confidence in their predictions. In the LITMUS++ phase, this dropped to 8.6%, while "very" surprised rose from 7.5% to 25.7%, indicating that the system improved participants' calibration confidence.

F.11 Workload and Final Survey

A mid-survey (after baseline) and final survey (after LITMUS++) collected NASA-TLX-style workload ratings. Mental demand (3.4 vs. 3.4), frustration (3.5 vs. 3.6), and success feeling (2.8 vs. 2.7) remained similar across phases, while physical effort increased slightly (3.2→3.6). In the final survey, participants rated the system on transparency (4.3/5), overall utility (3.9/5), ease of use (3.9/5), trust (3.7/5), and future use (3.7/5). Five participants indicated that LITMUS++ improved accuracy (3 "clearly," 2 "slightly"), while two rated it as equivalent.

F.12 Qualitative Feedback

The most valued aspects were deeper analysis with predictive models, use of linguistic features for regression, and clearer justification of predictions. The most common criticism was response latency (mentioned by 4 participants). One expert noted that LITMUS++ sometimes produced outputs that differed from other tools, requiring additional paper verification. Feature requests included faster response times, source hyperlinks, and broader general knowledge.

G Coder Agent Analysis

Figure 15 shows the regression algorithms selected by the Coder agent across tasks. The dominant algorithms are Random Forest, Ridge Regression, and XGBoost, which together account for the majority of model-fitting attempts. The algorithm mix varies by task: code generation and math reasoning favour ensemble methods, while text summarisation and machine translation show higher usage of linear models. This variation reflects the Coder agent's ability to adapt its modelling strategy to the characteristics of each task.

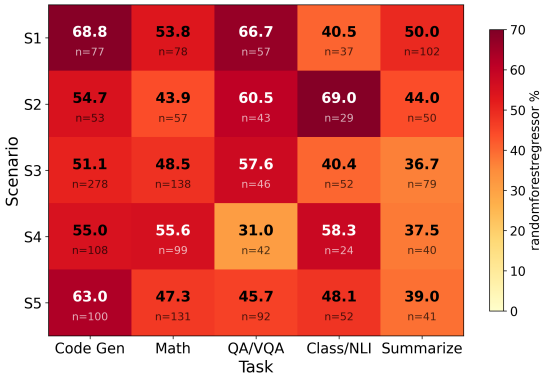


Figure 15: Regression algorithms selected by the Coder agent, broken down by task. Darker cells indicate higher usage frequency.

Figure 16 shows how `lang2vec` feature usage varies across scenarios. Usage is highest in Scenarios 3 and 4 (similar and distant language transfer), where typological features are most informative for cross-lingual regression. In Scenario 1, where direct evidence is available, the agent relies less on

typological features and more on retrieved numeric results.

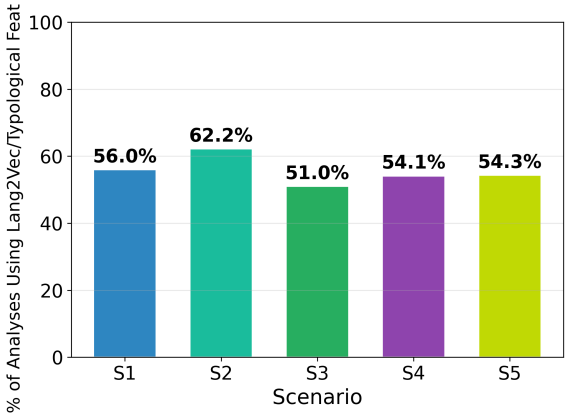


Figure 16: Usage of `lang2vec` typological features by scenario. Cross-lingual scenarios (S3, S4) show the highest feature utilisation.

Figure 17 provides a finer breakdown of `lang2vec` feature categories used by the Coder agent across tasks. Syntactic and geographic features are selected most frequently, with moderate use of phonological and genealogical signals. This pattern is consistent with feature-selection behaviour observed in cross-lingual transfer settings, where structural and areal similarity often provide stronger predictive cues than lexical overlap alone.

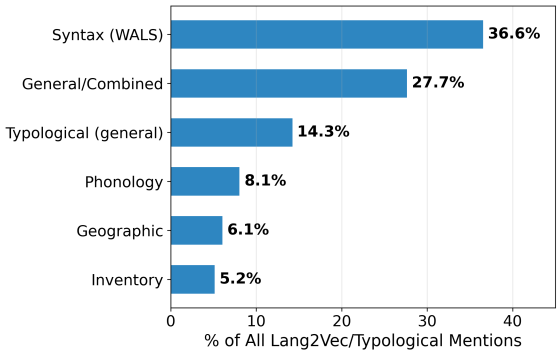


Figure 17: Breakdown of `lang2vec` feature categories used by the Coder agent. Syntactic and geographic features dominate across all tasks.

H Benchmark Creation

H.1 Scenario-wise Coverage

Figure 18 reports scenario-wise benchmark coverage used for question construction. The left panel shows the number of unique languages represented in each task–scenario block, while the right panel shows the number of unique model families. As expected, direct-evidence scenarios provide broader coverage, while transfer-based scenarios operate under narrower and more selective support.

H.2 Language Distribution Across Questions

Figure 19 reports language frequency across all benchmark questions. The distribution is strongly long-tailed: a small set of languages appears frequently, while many languages appear comparatively rarely. This reflects the benchmark design goal of preserving multilingual breadth while retaining sparse-evidence conditions common in low-resource evaluation.

H.3 Model Family Distribution Across Questions

Figure 20 reports model-family frequency across all benchmark questions. Similar to the language distribution (Figure 19), model-family coverage is long-tailed: a few widely benchmarked families dominate, while many others appear infrequently. This imbalance mirrors real-world evaluation practice, where popular model families receive disproportionate attention, and reinforces the need for transfer-based prediction in under-evaluated settings.

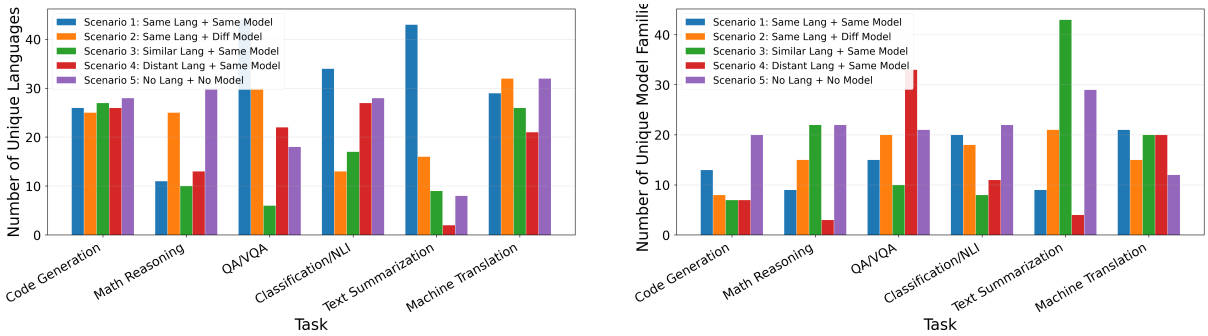
H.4 Language Similarity Computation Details

We determine whether two languages are similar using language feature vectors from the `lang2vec` library with feature set `syntax_knn+fam+geo`, which combines syntactic nearest-neighbour features, language family features, and geographic features. For languages with available vectors, we compute cosine distance between L2-normalised vectors, defined as $1 - \cos(\mathbf{u}, \mathbf{v})$.

We set the close-language threshold dynamically using the empirical distribution of pairwise distances. Specifically, we compute all pairwise cosine distances among languages with vectors and set the threshold to the 10th percentile, which yields 0.1069 in our benchmark construction. Language pairs at or below this threshold are treated as *close*, and pairs above this threshold are treated as *distant*. Scenario 3 selects close languages, while Scenario 4 explicitly excludes them.

H.5 Prompt Template for Question Generation

The prompt for question generation for PREDSET and QNASET is shown below.



(a) Number of unique languages used for question construction per task and scenario.

(b) Number of unique model families used for question construction per task and scenario.

Figure 18: Unique language and model-family coverage by task and scenario.

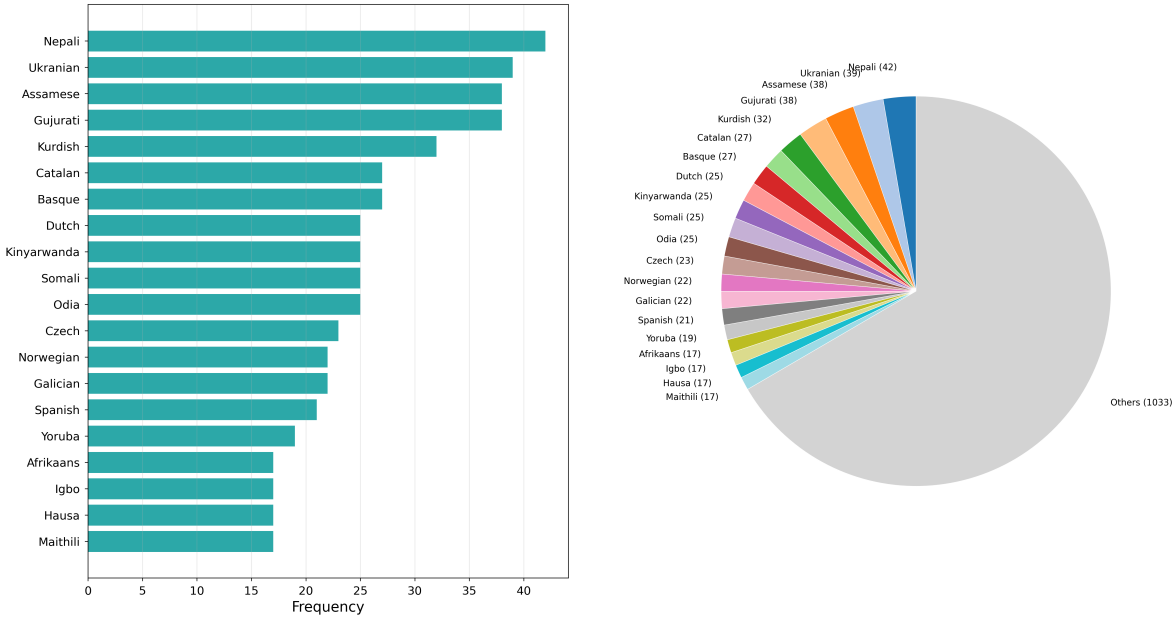


Figure 19: Language frequency across all benchmark questions. Left: top 20 languages by question frequency. Right: the same distribution with remaining languages grouped as Others.

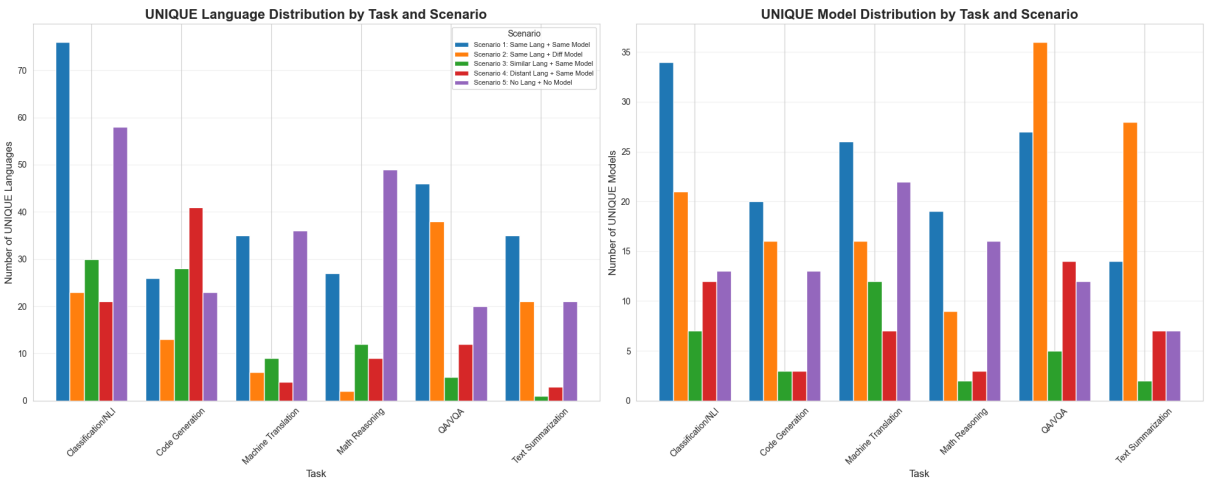


Figure 20: Model-family frequency across all benchmark questions. Left: top model families by question frequency. Right: the same distribution with remaining families grouped as Others.

```
You are given a language-to-models mapping for a specific NLP task where ground truth (GT)
    data is available.

Task: {task_name}
Language-Model Mappings: {mappings}

The mappings show which models have been benchmarked on which languages for this task.
Each language maps to a list of models that have been evaluated on it.

### Goal

Generate natural, varied evaluation questions for this task using the language-model
    combinations from the mapping.
Generate questions of the following types:

1. **Task + Model + Language**: Questions about specific model performance on specific language
2. **Task + Language**: Questions about how different models perform on a specific language
3. **Task + Model**: Questions about how a specific model performs across languages
4. **Model + Language**: Questions about model performance on language (task implied)
5. **Task only**: Questions about overall task performance across languages/models

### Question Templates (adapt and vary phrasing):

1. *What is the performance of {{model}} on {{task}} for {{language}}?*
2. *How does {{model}} perform on {{task}} in {{language}}?*
3. *Compare {{model1}} and {{model2}} performance for {{language}} on {{task}}.*
4. *Which model performs best for {{task}} in {{language}}?*
5. *What are the {{task}} results for {{model}} across all languages?*
6. *How does {{language}} performance vary across models for {{task}}?*
7. *What languages show the best performance for {{task}}?*
8. *Which models have been evaluated for {{task}} in {{language}}?*
9. *What is the cross-lingual performance of {{model}} on {{task}}?*
10. *How do low-resource languages perform on {{task}}?*

### Rules

1. **Only use languages and models from the provided mapping** - do not invent new ones.
    Generate questions only for natural languages. For tasks like code generation, programming
    languages might be present as well. DO NOT generate questions for programming languages.
2. Use the exact task name provided.
3. If multiple models/languages appear in a question, list them alphabetically,
    comma-separated (no spaces).
4. If task, model, or language is not present in a question, set it as `""` in the output.
5. Questions must be clear, end with `?`, and avoid hallucinations.
6. Generate diverse questions; avoid repetitive phrasing.
7. Focus on realistic evaluation scenarios where GT data would be available.

### Output Format

Return **only JSON** in this structure:

class QuestionsGenerated(BaseModel):
    class Question(BaseModel):
        complete_question: str
        task: str
        models: str
        languages: str

    questions: list[Question]

Generate {num_questions} diverse questions covering different question types and
    language-model combinations.
```

Figure 21: Prompt template used to generate illustrative questions for PREDSET (numeric prediction) and QNASET (comparative reasoning).

H.6 LLM-as-Judge Prompt for Quality Metrics

The following prompt is used to evaluate system responses on four quality dimensions: predictive plausibility, feature selection, coherence, and citation emphasis. Each response report is scored on a 1–5 scale by GPT-4.1 acting as an automated judge.

H.7 Prediction Extraction Prompt

After each system generates a response report, predictions are extracted using the following LLM prompt. This prompt parses the agent's report to produce structured predictions in a format compatible with ground-truth evaluation (MAE for numeric predictions, accuracy for comparative questions).

H.8 Thought Creator Evaluation Prompt

The following prompt evaluates whether hypotheses generated by the ThoughtCreatorAgent faithfully reflect expert guidance and stay within system capabilities. This produces the thought faithfulness and capability compliance metrics reported in Table 4.

H.9 Generated Code Evaluation Prompt

The following prompt evaluates the quality of Python code generated by the Coder agent. It assesses algorithm appropriateness, feature engineering, methodology rigor, code quality, and research alignment, producing the metrics reported in the Coder Agent Analysis appendix.

H.10 Tool Call Relevance Evaluation Prompt

The following prompt evaluates whether web search and crawl tool calls are relevant to the active hypothesis. This produces the web search relevance metric reported in Table 4.

I Use of AI Assistants

AI writing assistants were used during the preparation of this manuscript for paraphrasing and grammar correction. All scientific content, experimental design, analysis, and conclusions are the sole work of the authors.

```
You are an expert reviewer: a senior multilingual-NLP researcher and applied statistician.
Your job: take a single input -- a research report -- and produce an automatic, objective
    evaluation along four axes:
Predictive plausibility, Feature selection, Coherence, and Citation emphasis.
For each axis return an integer score 1-5 plus a short rationale and up to 3 concrete
    indicators.

Scoring rubric (1-5):

1) Predictive plausibility:
   5: Clear assumptions; modeling choices justified; uncertainty quantified; sanity checks
    present.
   4: Good justification; minor gaps (e.g., limited uncertainty discussion).
   3: Some plausible ideas but notable missing justifications or contradictory reasoning.
   2: Weak or ad-hoc predictive reasoning; unexplained leaps.
   1: Implausible claims, contradictory or absent reasoning.

2) Feature selection:
   5: Expert-level: linguistically justified features, selection method described, interaction
    effects explored.
   4: Strong feature list with reasonable justifications; some missing depth.
   3: Useful features but shallow rationale; key multilingual features missing.
   2: Generic or ill-suited features; little reasoning.
   1: No coherent feature rationale; features absent or irrelevant.

3) Coherence:
   5: Logical flow; claims follow from premises; clear definitions; few language errors.
   4: Mostly coherent; minor organizational or phrasing issues.
   3: Fragmented arguments or occasional contradictions; several clarity issues.
   2: Disorganized, hard to follow, or contradictory statements.
   1: Incoherent, contradictory, or unintelligible.

4) Citation emphasis:
   5: Key claims tied to relevant citations; literature justifies design choices.
   4: Many claims cited, but a few unsupported assertions remain.
   3: Some grounding, but important claims lack citations.
   2: Sparse citation support; many claims unsupported.
   1: Virtually no citation grounding; unusual claims without references.

Output format (strict JSON only):
{
  "metrics": [
    {"metric_name": "predictive_plausibility", "score": 5,
     "rationale": "...", "indicators": ["...", "...", "..."]},
    {"metric_name": "feature_selection", "score": 4,
     "rationale": "...", "indicators": ["..."]},
    {"metric_name": "coherence", "score": 3,
     "rationale": "...", "indicators": ["..."]},
    {"metric_name": "citation_emphasis", "score": 2,
     "rationale": "...", "indicators": ["..."]}
  ],
  "overall_recommendation": {
    "average_score": 3.50,
    "verdict": "Minor revision",
    "top_actionable_improvements": ["...", "...", "..."]
  }
}

Verdict mapping: >=4.25 Accept, 3.25-4.24 Minor revision,
2.5-3.24 Major revision, <2.5 Reject.
Rationales max 50 words. Indicators max 20 words each, up to 3 per metric.
If a metric cannot be assessed, set score to null with one-line rationale.

Report: {report}
```

Figure 22: LLM-as-judge prompt for evaluating system responses on four quality dimensions (predictive plausibility, feature selection, coherence, citation emphasis). Each metric is scored 1–5 with structured rationale and indicators.

```
Extract agent's predicted answer from research report.

USER QUERY: {user_query}
QUERY TYPE: {predictive_or_qna}

AGENT RESEARCH REPORT:
{agent_report}

INSTRUCTIONS:

1. DETERMINE is_answer_present:
   - Set to true ONLY if agent's report contains a relevant, clear answer
   - Set to false if no answer found or report is unclear/irrelevant

2. FOR PREDICTIVE QUERIES:
   - Extract ALL performance metrics mentioned (e.g., accuracy, pass@1, BLEU, ROUGE, F1)
   - For EACH metric found, create an object with:
     * metric_name: exact name of the metric (string)
     * value: original value AS STATED in the report (string)
     * value_in_100_range: numeric value scaled to 0-100 range (float)
   - Scaling rules for value_in_100_range:
     * If value is percentage (e.g., "61.25%"): extract number -> 61.25
     * If value is decimal 0-1 (e.g., "0.85"): multiply by 100 -> 85.0
     * If value already 0-100 (e.g., "85.5"): use as-is -> 85.5

3. FOR QNA QUERIES:
   - Extract CONCISE answer value(s) from agent's report:
     * For model questions: extract model name ONLY (e.g., "GPT-4")
     * For language questions: extract language name(s) ONLY
     * For numeric questions: extract number(s) ONLY
   - NEVER include full sentences or explanations

4. BOTH PREDICTIVE AND QNA:
   - If the question can be answered BOTH ways (metrics AND concise answer):
     * Provide BOTH predicted_metrics_and_values_for_predictive AND answer_text_for_qna

OUTPUT FORMAT (strict JSON, no markdown):

For PREDICTIVE:
{
  "is_answer_present": true,
  "predicted_metrics_and_values_for_predictive": [
    {"metric_name": "pass@1", "value": "61.25", "value_in_100_range": 61.25}
  ],
  "answer_text_for_qna": ""
}

For QNA:
{
  "is_answer_present": true,
  "predicted_metrics_and_values_for_predictive": [],
  "answer_text_for_qna": "GPT-4"
}

CRITICAL RULES:
- Use ONLY information from agent's report
- value_in_100_range must be NUMERIC ONLY (float, no % or strings)
- answer_text_for_qna must be CONCISE - NO full sentences
- predicted_metrics_and_values_for_predictive is always a LIST ([] if empty)
- answer_text_for_qna is always a STRING ("" if empty)
- Output valid JSON only
```

Figure 23: Prompt used to extract structured predictions from system response reports. The extracted predictions are compared against ground truth to compute PredSet MAE (numeric prediction) and QnASet accuracy (comparative reasoning).

```
--- Faithfulness Reflection ---
You are evaluating whether thought hypotheses correctly reflect reasoning guidance.

Expert Knowledge/Guidance (from user_proxy message):
{user_proxy_message}

Generated Thoughts:
{thought_paths}

For EACH thought, determine:
- Does the thought's method/approach align with the reasoning steps suggested in the expert
    knowledge?
- Does it follow the strategic guidance provided?

Respond with JSON:
{"evaluations": [{"thought_name": "<name>", "is_faithful": true/false, "explanation": "<brief
    reason>"}]}

Be strict: mark false if the thought significantly deviates from suggested approaches or
    ignores key guidance.

--- System Capability Adherence ---
PROHIBITED suggestions (mark as non-compliant if suggested):
- Fine-tuning models
- Creating new datasets
- Downloading models locally
- Training new models from scratch
- Accessing external APIs not already available

For EACH thought, check the "method" field:
- Does it suggest any PROHIBITED operations?

Respond with JSON:
{"evaluations": [{"thought_name": "<name>", "is_compliant": true/false, "violations":
    ["violation1", ...]}]}
```

Figure 24: Prompts used to evaluate ThoughtCreatorAgent output: faithfulness reflection (alignment with expert guidance) and system capability adherence (avoidance of prohibited operations).

```
You are an expert code reviewer specializing in data science, ML, and research methodology.

Context:
- Task Domain: {task_context}
- User Question: {user_question}

Python Code to Analyze:
{code_content}

Assess the following dimensions:

0. TASK-QUERY ALIGNMENT: Does the code address the user's research question?
1. ALGORITHMS & MODELS: Identify all algorithms; assess appropriateness and correctness.
2. FEATURES & VARIABLES: List features; assess engineering sophistication and validity.
3. METHODOLOGY: Identify approach; assess statistical rigor and evaluation strategy.
4. CODE QUALITY: Assess organization, error handling, and best practices.
5. RESEARCH APPROPRIATENESS: Is the methodology scientifically sound?
6. SOPHISTICATION: Rate as Basic, Intermediate, or Advanced.

Output JSON:
{
  "algorithms_used": ["..."], "algorithm_appropriateness":
    "appropriate|questionable|inappropriate",
  "features_used": ["..."], "feature_engineering_level": "none|basic|moderate|advanced",
  "methodology_type": "regression|classification|...", "methodology_rigor":
    "high|moderate|low",
  "code_quality": "excellent|good|fair|poor",
  "task_query_alignment": "well_aligned|moderately_aligned|poorly_aligned",
  "sophistication_level": "basic|intermediate|advanced",
  "overall_assessment": "excellent|good|fair|poor",
  "strengths": ["..."], "weaknesses": ["..."], "summary": "1-2 sentences"
}
```

Figure 25: Prompt used to evaluate generated Python code quality across seven dimensions: task alignment, algorithms, features, methodology, code quality, research soundness, and sophistication.

```
You are evaluating whether a web search/crawl tool call query is relevant
for answering a specific research thought/hypothesis.

Thought Name: {thought_name}

Thought Context:
Hypothesis: {hypothesis}
Method: {method}
Background: {background}

Tool Call Query: {tool_query}

A query is RELEVANT if:
- It directly searches for information needed to test the hypothesis
- It seeks data, benchmarks, or evidence mentioned in the thought's method
- It looks for papers, documentation, or results that would help answer the thought
- It searches for related concepts, models, or comparisons needed for analysis

A query is NOT RELEVANT if:
- It searches for completely unrelated topics
- It duplicates information already available in the thought context
- It's too vague or generic to be useful
- It searches for information not needed for this specific thought

Return JSON:
{
  "query": "{tool_query}",
  "thought_name": "{thought_name}",
  "is_relevant": true/false,
  "reasoning": "Brief explanation (1-2 sentences)"
}
```

Figure 26: Prompt used to evaluate web search and crawl tool call relevance against the active hypothesis context.