IPM-Bench: A Multi-task Benchmark for Evaluating Large Language Models in Integrated Pest Management

Anonymous ACL submission

Abstract

Integrated Pest Management (IPM) is essential for sustainable agriculture and global food security, yet its complexity presents significant challenges for AI support (Deguine et al., 2021; Zhou et al., 2024). To address the lack of evaluation tools for large language models (LLMs) in this domain, we present IPM-Bench, a comprehensive expert-curated multi-task benchmark tailored to real-world IPM scenarios. It comprises 2600 high-quality examples sourced from thousands of expert-authored documents published by leading U.S. university extension programs and global plant health knowledge resources (Sivapragasam and Chan, 2017). Covering 13 diverse task types including classification, question answering, summarization, and named entity recognition, the benchmark mirrors the full spectrum of IPM workflows from pest diagnosis to management decision making. We evaluated a broad range of frontier, open-source, and agriculture-oriented LLMs to establish performance baselines. While many models demonstrate strong general capabilities, they often struggle with complex reasoning and strict format adherence. IPM-Bench offers a critical resource for advancing and assessing AI systems for agricultural decision-making under workflow-oriented, evidence-grounded evaluation. We will release the benchmark and evaluation code to support reproducible research.

1 Introduction

Crop protection is a persistent bottleneck for stable food production. Pests and pathogens cause substantial global yield losses, and these risks are expected to intensify as climate change shifts pest ranges and outbreak dynamics (Savary et al., 2019; Deutsch et al., 2018). At the same time, resistance, non-target effects, and environmental externalities increase the need for effective and sustainable control strategies (Deguine et al., 2021).

Integrated Pest Management (IPM) (Elliott et al., 1995) addresses this need through monitoring, prevention, biological control, and selective chemical interventions guided by thresholds and local constraints (Kogan, 1998; Barzman et al., 2015). In practice, IPM requires more than rule memorization: practitioners must identify pests and symptoms, interpret trends and phenology, account for crop stage and weather, and translate evidence into timely actions. Yet IPM expertise and decision support remain uneven, and adoption is often limited by knowledge gaps, workflow friction, and trust or usability barriers (Parsa et al., 2014; Tonle et al., 2024; Marinko et al., 2025).

Large language models (LLMs) offer a potential path to scalable decision support, given their instruction following and multi-step reasoning capabilities (Achiam et al., 2023). However, standard NLP benchmarks rarely reflect IPM as a chain of grounded subtasks, and existing agriculture benchmarks emphasize adjacent skills rather than IPM workflows. For example, AgEval targets plant stress phenotyping (Arshad et al., 2024), AgXQA focuses on extension-style Q&A (Kpodo et al., 2024), CROP evaluates crop-science knowledge (Zhang et al., 2024a), and SeedBench centers on seed science (Ying et al., 2025). These benchmarks are valuable but do not directly test end-to-end IPM reasoning from situation understanding to actionable planning.

To bridge this gap, we introduce IPM-Bench, a multi-task benchmark that operationalizes IPM decision-making as a set of grounded competencies spanning pest and symptom identification, context-aware knowledge lookup, infestation or risk assessment, and integrated control recommendations with rationale. Scenarios are information-rich and paired with expert-validated answers to reflect real advisory constraints. We then benchmark a diverse set of proprietary, open-source, and agriculture-adapted LLMs on IPM-Bench, organized around three questions:

RQ1. What is the relationship between a model's general reasoning ability and its performance on complex IPM tasks?

RQ2. Do domain-specific or fine-tuned models outperform general LLMs on pest management problems?

RQ3. How does model scale trade off with domain competence on IPM-Bench tasks?

Answering these research questions will clarify current LLM capabilities and limitations in workflow-oriented IPM decision making, and more broadly inform how foundation models can be evaluated and deployed for AI-for-Science applications in agriculture. Our contributions are threefold:

- IPM-Bench is introduced as a purpose-built benchmark for evaluating LLMs on integrated pest management workflows.
- IPM-Bench covers key IPM competencies through diverse task formats, with expert-validated scenarios and question–answer pairs.
- IPM-Bench enables a broad empirical comparison across model families and scales, revealing strengths, failure modes, and promising directions for IPM-oriented LLM research.

2 Related Work

2.1 Domain-Specific LLM

General-purpose LLMs can act as strong generalists, but their reliability often decreases in specialized domains where terminology, constraints, and reasoning conventions are strict, motivating domain adaptation through continued pretraining and instruction tuning on in-domain corpora, frequently combined with grounding via retrieval or external tools. This pattern appears consistently across domains: in medicine, Med-PaLM targets clinical question answering and safety-oriented evaluation (Singhal et al., 2023b); in biomedicine, BioGPT emphasizes biomedical text generation and knowledge mining (Luo et al., 2022); in law, ChatLaw and Lawyer LLaMA adapt models to statutes and case-based reasoning with retrieval to reduce hallucinations (Cui et al., 2023; Huang et al., 2023); in finance, BloombergGPT is trained on large-scale financial text for domain-specific NLP workloads (Wu et al., 2023); in scientific writing, Galactica leverages scientific corpora to better handle technical content (Taylor et al., 2022); in code, Code Llama specializes in programming generation and infilling (Rozière et al., 2023); and in geospatial analysis, GeoGPT integrates LLM planning with GIS tools for multi-step workflows (Zhang et al., 2024b). Collectively, these studies suggest that effective domain LLMs are shaped not only by specialized data, but also by appropriate grounding mechanisms and domain-aligned evaluation.

2.2 Domain-Specific Benchmark

Rigorous evaluation is essential for validating domain adaptation, motivating a growing ecosystem of benchmarks tailored to domain constraints and real workflows. In agriculture, AgriBench evaluates multimodal agricultural reasoning (Zhou and Ryo, 2024), while CROP and SeedBench probe crop- and seed-science knowledge and decision processes (Zhang et al., 2024a; Ying et al., 2025). In medicine, MultiMedQA and related exam-style suites (e.g., MedMCQA) provide standardized tests for clinical QA and safety-sensitive answering (Singhal et al., 2023a; Pal et al., 2022). In law, LexGLUE targets diverse legal NLP tasks and LegalBench focuses on legal reasoning skills defined by legal experts (Chalkidis et al., 2022; Guha et al., 2023). In finance, FinanceBench evaluates open-book financial QA with evidence grounding (Islam et al., 2023). In geospatial practice, recent benchmarks move beyond factual QA toward multistep, tool-oriented geospatial tasks (Zhang et al., 2025). For structured-knowledge settings, KG-LLM-Bench and the LLM-KG-Bench framework benchmark KG understanding and semantic-technology tasks (Markowitz et al., 2025; Meyer et al., 2025). Finally, meta-suites such as BenchHub help aggregate and slice evaluations by skill and domain (Kim et al., 2025), and surveys further systematize domain-specific benchmark design for multi-modal LLMs (Anjum et al., 2025).

3 IPM-Bench

This section presents the design principles of IPM-Bench and details its 13 evaluation task types. IPM-Bench systematically evaluates LLMs for integrated pest management by aligning with the typical workflow followed by practitioners. The benchmark is organized into four primary categories: pest identification, knowledge retrieval, infestation assessment and reasoning, and control strategy recommendation. This taxonomy ensures comprehensive coverage of domain knowledge and decision skills.

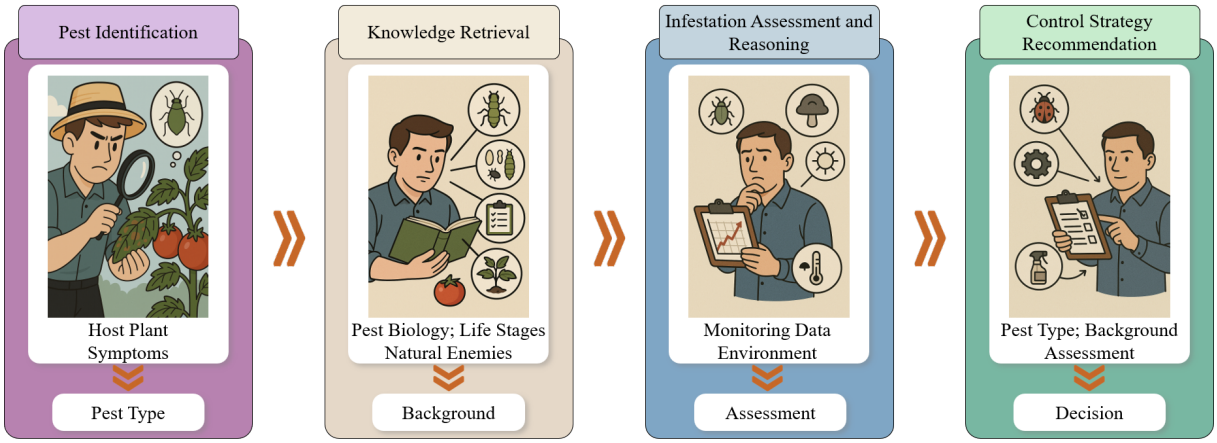

Figure 1: Integrated Pest Management Workflow Framework. We depict four canonical stages used by practitioners to make field decisions: (1) Pest Identification; (2) Knowledge Retrieval; (3) Infestation Assessment and Reasoning; (4) Control Strategy Recommendation. See Appendix.A.1 for details.

3.1 IPM Process Overview

IPM practitioners typically follow four key steps in field decision making (Figure 1): identify and monitor pests, determine whether observed levels exceed action/economic thresholds, apply prevention and control tactics that minimize risk, and evaluate outcomes for continuous improvement (Zhou et al., 2024; Leach et al., 2025). These steps form the basis for IPM-Bench's task categorization.

3.2 Task Taxonomy

Tasks in IPM-Bench are grouped into four main areas corresponding to the IPM workflow. Each area is further divided into subcategories to provide a systematic evaluation of LLM capabilities across recognition, retrieval, reasoning and recommendation (Figure 1). The categorization was developed with input from extension specialists and IPM domain experts.

3.2.1 Pest Identification

LLMs detect and recognize pest entities and map field descriptions to likely causal agents. Specific tasks include:

- Pest Named Entity Recognition (DS-3)
- Crop–Pest / Symptom–Pest Matching (DS-2)
- Pest Identification, multiple choice (QA-1)
- Pest Identification, multi answer (QA-2)

3.2.2 Knowledge Retrieval

LLMs retrieve essential knowledge needed before intervention, including pest biology, life stages, host range, natural enemies and preventive practices. Specific tasks include:

- Cloze facts on pest biology (QA-3)
- Open factual generation (QA-4)

3.2.3 Infestation Assessment and Reasoning

LLMs analyze monitoring data and context to decide whether intervention is warranted and to justify that decision. Specific tasks include:

- Threshold decision, multiple choice (RC-1)
- Scenario reasoning, multi answer (RC-2)
- Numeric threshold cloze (RC-3)
- Open-ended reasoning (RC-4)
- Action classification (DS-1)

3.2.4 Control Strategy Recommendation

LLMs compose coherent, low-risk management plans and summarize key steps for record keeping and evaluation. Specific tasks include:

- IPM plan generation (SUM-1)
- Key-information extraction from plans (SUM-2)

3.3 Benchmark Construction

3.3.1 Data Collection

We built IPM-Bench from authoritative, extension-grade IPM literature and global plant health knowledge resources. Using a focused crawler, we harvested 3,000+ PDF documents from five primary sources: Cornell University's Integrated Pest Management program[1], the UC Statewide IPM Program[2], Texas A&M AgriLife Extension's IPM program[3], University of Minnesota Extension IPM

[1] https://cals.cornell.edu/integrated-pest-management

[2] https://ipm.ucanr.edu/

[3] https://agrilifeextension.tamu.edu/assets/insects-pests-diseases/pesticides-pest-management/integrated-pest-management/

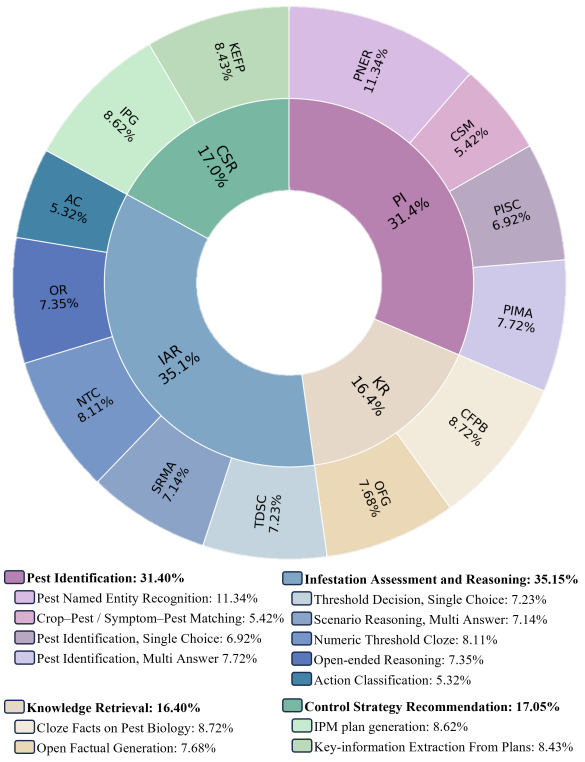

Figure 2: Benchmark Taxonomy Distribution. IPM-Bench organizes the IPM workflow into four major stages: Pest Identification, Knowledge Retrieval, Infestation Assessment and Reasoning, and Control Strategy Recommendation. Each stage is further divided into expert-curated subcategories. This figure reports the distribution of the 90,259 automatically generated candidate items (Section 3.3.2), from which the final 2,600-item benchmark is curated (Section 3.3.3). Percentages denote the share of items in each category relative to 90,259.

resources[4], and CABI PlantwisePlus Knowledge Bank[5]. These sites collectively provide diagnostic guides, monitoring and threshold recommendations, and practice-oriented management advice that reflect standard IPM workflows and terminology.

We converted PDFs to text, segmented the text into sentences, and applied a multi-stage cleaning pipeline. Heuristics removed boilerplate, repeated headers/footers, tables of contents, and sentences shorter than 6 tokens. We then performed near-duplicate removal to prevent over-representation of templated guidance across documents; deduplication is known to improve downstream modeling and evaluation reliability in large text corpora (Lee et al., 2022) (details in Appendix.B.1). Finally, we filtered to sentences that contain at least one crop, pest, or management keyword drawn from curated lexicons (hosts, life stages, natural enemies, tactics). This yielded 6,943 high-quality sentences that serve as atomic facts or context snippets for downstream generation.

To preserve the original intent of extension materials and avoid unsafe content, we excluded sentences that solely prescribe application rates or restricted-use pesticide instructions without surrounding context, and we kept the focus on identification, monitoring, thresholds, preventive tactics, and integrated recommendations.

3.3.2 Automatic Question Generation

From 6,943 high-quality sentences, we used Gemini-2.5-Pro to synthesize candidate items across 13 task types that mirror the canonical IPM workflow (Comanici et al., 2025). We generated approximately 90,259 candidate items in total, with prompts requiring JSON-formatted outputs and emphasizing evidence grounding, format adherence, and IPM-safe recommendations. Task construction followed type-specific rules: reading-comprehension items were written from short passages composed of one or more cleaned sentences; fill-in-the-blank items masked salient spans (e.g., life stage, host, threshold value) and required exact reference answers; and strategy items added minimal scenario scaffolding (crop, growth stage, constraints) to elicit concise, multi-tactic IPM plans consistent with extension guidance (details in Appendix.B.2).

3.3.3 Quality Validation

To reduce noise while preserving a large candidate pool, we first applied an LLM-as-judge filtering stage: a separate evaluator model scored each item for answerability, faithfulness to the cited evidence, format adherence, and safety or IPM compliance (Liu et al., 2023; Zheng et al., 2023) (details in Appendix.C.1). This stage is used only to rank noisy candidates; final selection is determined by automatic checks and IPM experts to mitigate evaluator bias. For each task type, we retained the top-300 highest-scoring items (300×13 = 3,900) for downstream verification. We then conducted a two-round validation that combines automatic checks with domain-expert review. In Round 1 (Automatic Validation), we ran exact-match recoverability for cloze and choice items, overlap consistency checks for short-answer items, schema

[4] https://vegedge.umn.edu/
[5] https://plantwiseplusknowledgebank.org/

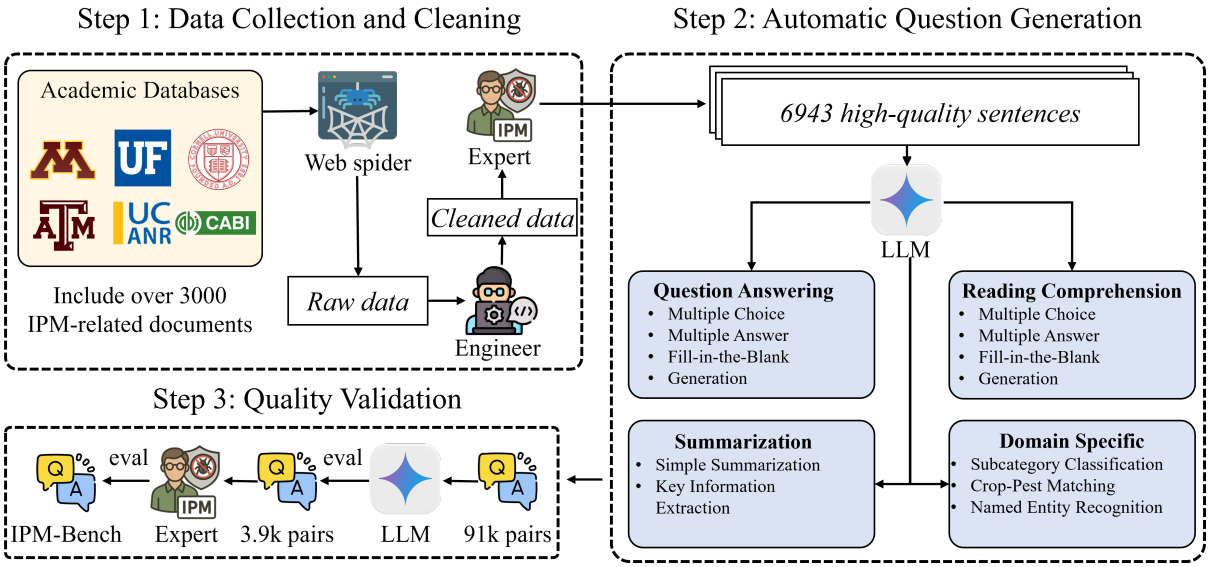

Figure 3: Benchmark Construction Pipeline. We crawl extension repositories and plant health knowledge resources, convert and clean documents, deduplicate sentences, synthesize candidate questions, rank candidates using an LLM evaluator, and curate the final benchmark with IPM experts.

Type ID	Question Type	Metric	Count (n)
Q&A			
QA-1	multiple choice	Accuracy	200
QA-2	Multiple Answer	Macro-F1	200
QA-3	Fill-in-the-Blank	ROUGE-L	200
QA-4	Generation	ROUGE-L	200
Reading Comprehension			
RC-1	multiple choice	Accuracy	200
RC-2	Multiple Answer	Macro-F1	200
RC-3	Fill-in-the-Blank	ROUGE-L	200
RC-4	Generation	ROUGE-L	200
Summarization			
SUM-1	Simple Summarization	ROUGE-L	200
SUM-2	Key Information Extraction	ROUGE-L	200
Domain-Specific			
DS-1	Simple Choice	Accuracy	200
DS-2	Simple Choice	Accuracy	200
DS-3	Key Information Extraction	Macro-F1	200

Table 1: Benchmark Task Types. Each high-quality text segment systematically covers one of the 13 distinct task types to ensure comprehensive IPM workflow coverage. Count (n) indicates the number of questions retained for that task type after quality validation, in the final benchmark (2,600 total) in IPM-Bench. The complete distribution statistics are provided in Appendix.C.

parsing and field completeness for structured outputs, and near-duplicate removal, yielding 3,158 validated candidates (details in Appendix.C.2.1). In Round 2 (Expert Review), two IPM experts reviewed the validated pool and selected or refined 200 items per task type, resolving ambiguities and ensuring realism and alignment with extension guidance. The final benchmark contains 2,600 items (200×13), and Table 1 reports the task taxonomy, evaluation metric, and per-type counts (details in Appendix.C.2.2).

3.4 Evaluation

Our evaluation has two stages: (i) response standardization and (ii) task-specific scoring. Table 1 summarizes the 13 task types in IPM-Bench, along with the evaluation metric and the number of instances per type.

- Multiple Choice Tasks: Accuracy.
- Multiple Answer tasks: Macro-F1.
- Fill-in-the-Blank: ROUGE-L F1 (Lin, 2004).
- Generation Tasks: ROUGE-L F1. BERTScore (Zhang et al., 2019) included as an additional evaluation metric (Appendix.D.2).

Formal mathematical definitions and post-processing methods are detailed in Appendix.D. To account for prompt sensitivity in LLMs, Appendix.F presents all prompt templates used across tasks and models,

4 Experiment

4.1 Experimental Setup

Models We evaluate 24 large language models spanning three groups to reflect realistic deployment choices. The proprietary group includes frontier general-purpose systems (GPT-

4o[6], GPT-4o mini[7], GPT-3.5 Turbo[8], Gemini variants[9], and GLM-4 series[10]). The open-source group covers widely used families at different scales (Llama[11], Qwen[12], InternLM[13], Mistral[14], and GLM-4-Chat[15]), enabling controlled comparisons between small and large checkpoints. The domain-specific group includes agriculture-adapted models (CropSeek[16], Aksara-v1-7B[17], PLLaMa-7B/13B[18]), which allows us to test whether domain adaptation improves IPM workflow performance relative to general models (Yang et al., 2024).

Implementation Details All models are queried with standardized, format-constrained prompts (Appendix.F). Each prompt specifies the required output schema for the task, and we keep prompt structure consistent across models to limit prompt-induced variance. Model outputs are parsed using task-aware rules and normalized into comparable representations, such as single labels for multiple-choice, sets of labels for multiple-answer tasks, and extracted spans for fill-in-the-blank items. We then compute task-appropriate metrics (accuracy, Macro-F1, ROUGE-L F1), with additional implementation details, templates, and inference settings reported in the Appendix.D.

4.2 Performance Evaluation

Table 2 summarizes results across 13 task types and the overall average (AVR). Proprietary models lead overall, with GPT-4o achieving the best AVR (67.90) and Gemini-2.0-Flash close behind (66.97), while GPT-4o mini and Gemini-1.5-Pro remain in the mid-60s and GPT-3.5 Turbo still exceeds most open-source systems on average. Open-source models form a middle tier where larger checkpoints perform best, such as Qwen2.5-72B (53.72) and Qwen2-72B (53.31), and many 7B to 14B models stay below 50. Domain-specific models are mixed: Aksara-v1-7B (49.40) is comparable to mid-tier open models, but CropSeek (27.49) and PLLaMa-7B (19.50) lag substantially, indicating that domain adaptation alone does not ensure robust IPM workflow competence. Performance also varies by task format: constrained classification is generally easier, whereas open-ended generation (QA-4 and RC-4) and summarization (SUM-1 and SUM-2) remain difficult even for top systems, and near-zero scores on fill-in-the-blank tasks (QA-3 and RC-3) despite strong multiple-choice results highlight the importance of exact-span extraction and strict format compliance. Overall, IPM-Bench separates basic recognition from workflow-level reasoning and grounded decision support, providing a more diagnostic measure of practical readiness. Error Analysis of Large Language Model Outputs is provided in Appendix.E.

4.3 Empirical Analysis

4.3.1 Analysis on Reasoning Ability

Across IPM-Bench, reasoning-intensive tasks widen the gap far more than recognition-oriented ones. Many models, especially proprietary, perform strongly on constrained classification tasks such as RC-1 and DS-1, yet struggle with multi-step synthesis, constraint satisfaction, and longer structured outputs. Even top systems remain challenged by open-ended generation (QA-4/RC-4) and summarization (SUM-1/2), indicating persistent bottlenecks in faithful, evidence-grounded generation. A complementary signal is robustness to strict rubrics. Fill-in-the-blank tasks (QA-3/RC-3) require exact-span reproduction and schema compliance, where several open-source and domain-adapted models score near zero despite reasonable QA-1/RC-1 performance, suggesting that formatting errors, omissions, or non-extractive paraphrases often drive failures rather than pure misunderstanding. Proprietary models are markedly more stable, with high RC-3 and strong RC-2, which lifts overall AVR. Together, these patterns imply that IPM workflow competence depends not only on knowledge and reasoning, but also on controllability, including reliable schema adherence and grounding in provided evidence. This echoes prior findings that general benchmark strength does not guarantee robust rubric-following under constrained evaluation protocols (Liang et al., 2022; Zheng et al., 2023). See Appendix.Gfor detailed per-task and per-model comparisons.

[6] https://platform.openai.com/docs/models/gpt-4o
[7] https://platform.openai.com/docs/models/gpt-4o-mini
[8] https://platform.openai.com/docs/models/gpt-3.5-turbo
[9] https://ai.google.dev/gemini-api/docs/models
[10] https://open.bigmodel.cn/dev/api
[11] https://www.llama.com/
[12] https://github.com/QwenLM/Qwen
[13] https://internlm.intern-ai.org.cn/
[14] https://mistral.ai/models
[15] https://open.bigmodel.cn/dev/api
[16] https://huggingface.co/persadian/CropSeek-LLM
[17] https://huggingface.co/cropinailab/aksara_v1
[18] https://arxiv.org/pdf/2401.01600

Model	QA-1	QA-2	QA-3	QA-4	RC-1	RC-2	RC-3	RC-4	SUM-1	SUM-2	DS-1	DS-2	DS-3	AVR
Proprietary LLMs														
Gemini-1.5-Pro	80.50	79.67	53.13	23.99	97.50	83.79	87.59	48.12	28.76	19.62	100.00	94.50	48.62	65.10
Gemini-2.0-Flash	83.00	82.70	57.04	27.33	97.00	97.12	94.25	48.58	29.06	9.28	100.00	95.50	49.75	66.97
Gemini-2.5-Flash_Lite	81.50	77.21	44.44	26.07	96.50	91.25	90.37	52.92	30.45	5.36	100.00	93.50	52.36	64.76
GPT-3.5Turbo	75.00	82.19	42.32	27.68	96.00	96.20	80.43	50.68	19.94	19.50	68.50	92.00	51.73	61.71
GPT-4o	85.00	86.45	53.55	24.23	96.50	98.04	92.47	51.80	26.76	22.37	99.50	95.50	50.50	67.90
GPT-4o mini	78.50	85.09	40.73	22.35	98.50	97.34	89.16	46.88	27.44	25.95	99.00	94.00	45.65	65.43
GLM-4-Plus	83.00	79.31	41.89	22.76	97.50	91.13	86.96	46.89	29.63	13.90	99.50	96.00	53.44	64.76
GLM-4.5	82.00	70.65	53.24	17.20	96.00	72.48	91.47	38.73	24.80	30.42	99.50	93.00	47.71	62.86
Open-Source LLMs														
GLM-4-Chat-9B	77.00	72.15	0.55	13.67	94.50	71.14	2.83	22.79	19.33	27.30	99.50	91.50	44.26	48.96
InternLM2.5-7B	70.50	76.36	13.53	18.86	95.50	91.97	38.70	38.53	12.09	7.87	99.50	90.50	34.09	52.92
InternLM3-8B	73.00	80.94	3.64	15.75	94.00	72.50	13.74	29.09	22.68	7.21	99.50	92.00	28.29	48.64
Llama3.1-8B	69.50	65.11	0.62	13.77	91.00	51.42	2.49	21.90	14.95	9.42	94.50	86.50	31.18	42.49
Llama3.1-70B	81.00	79.93	7.53	19.24	96.50	79.56	24.11	33.20	16.04	7.81	98.50	91.50	31.03	51.23
Mistral-v0.3-7B	72.00	81.48	4.56	19.84	88.50	95.28	8.99	42.52	21.04	12.74	98.50	90.00	25.92	50.87
Qwen2-7B	24.50	0.00	9.01	16.90	77.00	10.58	11.57	30.91	17.28	18.58	91.50	87.00	28.02	32.53
Qwen2-72B	80.50	79.96	12.95	15.10	96.50	79.24	45.13	28.55	17.82	18.98	99.00	94.00	25.35	53.31
Qwen2.5-7B	73.00	78.68	0.72	14.70	96.00	74.78	6.21	27.73	19.14	21.94	100.00	93.00	30.22	48.93
Qwen2.5-14B	83.00	79.04	0.68	14.85	97.00	76.43	2.16	24.34	16.90	25.92	99.00	94.50	34.53	49.87
Qwen2.5-72B	85.00	78.90	7.21	17.21	97.50	83.72	17.54	39.90	24.30	16.82	99.50	94.50	36.27	53.72
Qwen3-8B	43.00	75.92	0.66	12.89	65.00	70.36	2.24	20.77	11.58	1.77	84.50	71.50	0.00	35.40
Domain Specific LLMs														
CropSeek	17.50	77.58	0.32	11.89	32.50	65.05	2.94	21.90	13.41	0.81	68.00	37.50	0.00	27.49
Aksara-v1-7B	72.50	77.87	11.52	19.58	87.00	90.55	21.73	43.36	19.57	9.96	99.50	89.00	0.00	49.40
PLLaMa-7B	0.00	66.61	0.49	0.79	36.50	59.00	1.07	4.44	2.60	0.00	46.50	35.50	0.00	19.50
PLLaMa-13B	60.00	68.93	0.00	4.41	81.50	71.37	0.82	15.96	2.21	0.00	90.50	86.50	0.00	37.17

Table 2: Evaluation of 24 Large Language Models on the IPM-Bench benchmark. Performance is stratified across 13 distinct task types relevant to Integrated Pest Management. The columns delineate these tasks: Question Answering, including single-choice (QA-1), multi-choice (QA-2), fill-in-the-blank (QA-3), and open generation (QA-4); Reading Comprehension, with single-choice (RC-1), multi-choice (RC-2), fill-in-the-blank (RC-3), and open generation (RC-4); Summarization, covering simple (SUM-1) and key info extraction (SUM-2); and domain-specific tasks, namely subtask classification (DS-1), crop-pest matching (DS-2), and agricultural named entity recognition (DS-3). The final column (AVR) reports the overall average score across all tasks. The top five performers in each column are highlighted with a red color scale, where a darker shade indicates a higher rank.

4.3.2 Impact of Domain-Specific Fine-Tuning

Domain adaptation yields mixed gains on IPM-Bench. Aksara-v1-7B is relatively competitive and roughly matches mid-tier open-source models, suggesting that agriculture-oriented training can help with terminology and decision cues. However, CropSeek and PLLaMa variants lag and often fail on schema-sensitive tasks, indicating that domain adaptation alone is insufficient for reliable workflow execution.

Overall, the main bottleneck is structured-output controllability. Small formatting mistakes, missing fields, or non-extractive paraphrases can sharply reduce scores even when the model has relevant knowledge. This matches prior findings that parameter-efficient tuning can boost in-domain recall, while robust schema compliance and faithful, grounded generation usually require targeted instruction tuning and objective design.

4.3.3 Impact of Model Size

Figure 4 shows a clear but imperfect scaling pattern: average performance generally increases with model size on a log scale, and within major open-source families larger checkpoints tend to outperform smaller ones, consistent with compute-optimal training analyses (Hoffmann et al., 2022). However, size alone does not determine IPM-Bench results: small models vary widely, suggesting that instruction following, data quality, and alignment strongly affect robustness under strict schemas, and even the best open 70B to 72B models remain well below frontier proprietary systems, indicating headroom on workflow-level abilities such as multi-step reasoning, concise abstraction, and faithful extraction or generation under constrained output formats. Notably, the dispersion among similarly sized models implies that scaling benefits are often gated by controllability and evaluation-aware training rather than parameter count alone.

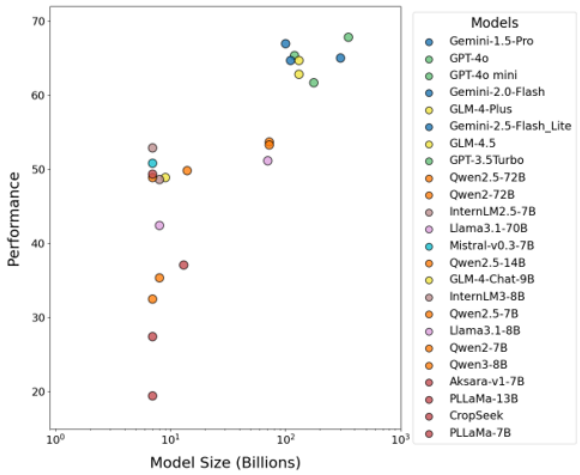

Figure 4: Performance vs. Model Size. Each point represents a model, plotting its overall IPM-Bench average score (AVR) against parameter count (in billions, log scale). Larger models generally achieve higher performance, but substantial variance remains among similarly sized models and a clear gap persists between open-source and frontier proprietary systems.

5 Discussion

IPM-Bench reveals a persistent gap between strong performance on constrained recognition and weaker performance on workflow-level reasoning and synthesis. Many models succeed on single-choice and basic reading comprehension, but often fail when exact-span extraction and strict schemas are required, when multiple contextual constraints must be integrated, or when concise, evidence-grounded recommendations are needed. This indicates that practical IPM assistants require not only domain knowledge, but also robust instruction following, controllable formatting, and faithful grounding.

Although larger models generally perform better, Figure 4 and Table 2 show that scale alone does not remove key failure modes. Similarly sized models vary widely, and the gap between the best open models and frontier proprietary systems suggests substantial headroom in multi-step reasoning and evidence-faithful generation. These findings motivate IPM-oriented designs that emphasize retrieval grounding from extension resources, structured tool use for rule and threshold checks, and calibrated uncertainty when context is insufficient, and IPM-Bench offers a controlled testbed to measure and diagnose such improvements.

6 Conclusion

This paper introduces IPM-Bench, a multi-task benchmark designed to evaluate LLMs on integrated pest management workflows. IPM-Bench is constructed from extension-grade IPM sources, covers 13 task types spanning identification, knowledge retrieval, infestation assessment, and control strategy recommendation, and includes expert-reviewed items to support reliable evaluation. Benchmarking 24 models shows that proprietary frontier models achieve the strongest overall performance, open-source models form a middle tier with clear benefits from scaling, and domain-adapted models yield mixed gains. Across all families, strict-schema extraction, grounded generation, and concise summarization remain key bottlenecks. IPM-Bench offers a practical testbed for future research on grounded, controllable, and domain-reliable LLMs for agricultural decision support.

7 Limitations

IPM-Bench has several limitations. It is mainly derived from English extension-style sources and may under-represent other regions, crops, and practices. It targets text-only, single-turn reasoning, and does not cover multimodal inputs or interactive decision making over time. Although expert review is used, parts of the dataset are automatically generated and filtered, so ambiguity and metric bias may remain, especially for ROUGE-style evaluation. Finally, results are obtained under standardized prompts and fixed model versions, and may shift under real deployment conditions or as models and prompting evolve.

8 Ethical Considerations

IPM guidance can affect human health, environmental safety, and regulatory compliance. IPM-Bench emphasizes identification, monitoring, thresholds, preventive tactics, and integrated recommendations, while excluding items that merely prescribe pesticide application rates or restricted-use instructions without context. It is intended for evaluation and research, not as a standalone decision authority. We release the benchmark items and metadata; evidence snippets are kept short and linked to original sources. Real deployments should include human oversight, cite sources, reflect local regulations, and communicate uncertainty, and we recommend pairing accuracy with safety-oriented evaluations to reduce misuse.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Khizar Anjum, Muhammad Arbab Arshad, Kadhim Hayawi, Efstathios Polyzos, Asadullah Tariq, Mohamed Adel Serhani, Laiba Batool, Brady Lund, Nishith Reddy Mannuru, Ravi Varma Kumar Bevara, and 1 others. 2025. Domain specific benchmarks for evaluating multimodal large language models. *arXiv preprint arXiv:2506.12958*.

Muhammad Arbab Arshad, Talukder Zaki Jubery, Tirtho Roy, Rim Nassiri, Asheesh Singh, Arti Singh, Chinmay Hegde, Baskar Ganapathysubramanian, Aditya Balu, Adarsh Krishnamurthy, and 1 others. 2024. Ageval: A benchmark for zero-shot and few-shot plant stress phenotyping with multimodal llms.

Marco Barzman, Paolo Bàrberi, A. Nicholas E. Birch, Piet Boonekamp, Silke Dachbrodt-Saaydeh, Benno Graf, Bernd Hommel, Jens Erik Jensen, and 1 others. 2015. Eight principles of integrated pest management. *Agronomy for Sustainable Development*, 35:1199–1215.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. Lexglue: A benchmark dataset for legal language understanding in english. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Jiaxi Cui, Munan Ning, Zongjian Li, Bohua Chen, Yang Yan, Hao Li, Bin Ling, Yonghong Tian, and Li Yuan. 2023. Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model. *arXiv preprint arXiv:2306.16092*.

Jean-Philippe Deguine, Jean-Noël Aubertot, Rica Joy Flor, Françoise Lescourret, Kris AG Wyckhuys, and Alain Ratnadass. 2021. Integrated pest management: good intentions, hard realities. a review. *Agronomy for Sustainable Development*, 41(3):38.

Curtis A Deutsch, Joshua J Tewksbury, Michelle Tigchelaar, David S Battisti, Scott C Merrill, Raymond B Huey, and Rosamond L Naylor. 2018. Increase in crop losses to insect pests in a warming climate. *Science*, 361(6405):916–919.

NC Elliott, JA Farrell, AP Gutierrez, Joop C van Lenteren, MP Walton, and Steve Wratten. 1995. *Integrated pest management*. Springer Science & Business Media.

Neel Guha, Julian Nyarko, Daniel Ho, Christopher Ré, Adam Chilton, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel Rockmore, Diego Zambrano, and 1 others. 2023. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *Advances in neural information processing systems*, 36:44123–44279.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Quzhe Huang, Mingxu Tao, Chen Zhang, Zhenwei An, Cong Jiang, Zhibin Chen, Zirui Wu, and Yansong Feng. 2023. Lawyer llama technical report. *arXiv preprint arXiv:2305.15062*.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. Financebench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Eunsu Kim, Haneul Yoo, Guijin Son, Hitesh Patel, Amit Agarwal, and Alice Oh. 2025. Benchhub: A unified benchmark suite for holistic and customizable llm evaluation. *arXiv preprint arXiv:2506.00482*.

Marcos Kogan. 1998. Integrated pest management: historical perspectives and contemporary developments. *Annual review of entomology*, 43(1):243–270.

Josué Kpodo, Parisa Kordjamshidi, and A Pouyan Nejadhashemi. 2024. Agxqa: A benchmark for advanced agricultural extension question answering. *Computers and Electronics in Agriculture*, 225:109349.

Ashley Leach, Arnol Ariel Gomez, and Ian Kaplan. 2025. Threshold-based management reduces insecticide use by 44% without compromising pest control or crop yield. *Communications Earth & Environment*, 6(1):710.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445.

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*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. Biogpt: Generative pre-trained transformer for biomedical text generation and mining. *arXiv preprint arXiv:2210.10341*.

Jurij Marinko, Vladimir Kuzmanovski, Mark Ramsden, and Marko Debeljak. 2025. Overcoming barriers to the adoption of decision support systems in integrated pest management in some european countries. *Agronomy*, 15(2):426.

Elan Markowitz, Krupa Galiya, Greg Ver Steeg, and Aram Galstyan. 2025. Kg-llm-bench: A scalable benchmark for evaluating llm reasoning on textualized knowledge graphs. *arXiv preprint arXiv:2504.07087*.

Lars-Peter Meyer, Johannes Frey, Desiree Heim, Felix Brei, Claus Stadler, Kurt Junghanns, and Michael Martin. 2025. Llm-kg-bench 3.0: A compass for semantic technology capabilities in the ocean of llms. In *European Semantic Web Conference*, pages 280–296. Springer.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

Soroush Parsa, Stephen Morse, Alejandro Bonifacio, Timothy C. B. Chancellor, Bruno Condori, Verónica Crespo-Pérez, Shaun L. A. Hobbs, Jürgen Kroschel, and 1 others. 2014. Obstacles to integrated pest management adoption in developing countries. *Proceedings of the National Academy of Sciences*, 111(10):3889–3894.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Serge Savary, Laetitia Willocquet, Sarah Jane Pethybridge, Paul Esker, Neil McRoberts, and Andy Nelson. 2019. The global burden of pathogens and pests on major food crops. *Nature ecology & evolution*, 3(3):430–439.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, and 1 others. 2023a. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Karan Singhal, Shekoofeh Azizi, Tao Tu, Shervin S. Mahdavi, and 1 others. 2023b. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

A Sivapragasam and Fook Wing Chan. 2017. Knowledge management resources in cabi for underutilized crops. *Regional Expert Consultation on Underutilized Crops for Food and Nutritional Security in Asia and the Pacific–Thematic, Strategic Papers and Country Status Reports*, page 82.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Franck B. N. Tonle, Saliou Niassy, Milliam M. Z. Ndadji, Maurice T. Tchendji, Armand Nzeukou, Bester T. Mudereri, Kennedy Senagi, and Henri E. Z. Tonnang. 2024. A road map for developing novel decision support system (dss) for disseminating integrated pest management (ipm) technologies. *Computers and Electronics in Agriculture*, 217:108526.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Xianjun Yang, Junfeng Gao, Wenxin Xue, and Erik Alexandersson. 2024. Pllama: An open-source large language model for plant science. *arXiv preprint arXiv:2401.01600*.

Jie Ying, Zihong Chen, Zhefan Wang, Wanli Jiang, Chenyang Wang, Zhonghang Yuan, Haoyang Su, Huanjun Kong, Fan Yang, and Nanqing Dong. 2025. Seedbench: A multi-task benchmark for evaluating large language models in seed science. *arXiv preprint arXiv:2505.13220*.

Hang Zhang, Jiawei Sun, Renqi Chen, Wei Liu, Zhonghang Yuan, Xinzhe Zheng, Zhefan Wang, Zhiyuan Yang, Hang Yan, Hansen Zhong, and 1 others. 2024a. Empowering and assessing the utility of large language models in crop science. *Advances in Neural Information Processing Systems*, 37:52670–52722.

Qianheng Zhang, Song Gao, Chen Wei, Yibo Zhao, Ying Nie, Ziru Chen, Shijie Chen, Yu Su, and Huan Sun. 2025. Geoanalystbench: A geoai benchmark for assessing large language models for spatial analysis workflow and code generation. *Transactions in GIS*, 29(7):e70135.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yifan Zhang, Cheng Wei, Zhengting He, and Wenhao Yu. 2024b. Geogpt: An assistant for understanding and processing geospatial tasks. *International*

Journal of Applied Earth Observation and Geoinformation, 131:103976.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Wentao Zhou, Yashwanth Arcot, Raul F Medina, Julio Bernal, Luis Cisneros-Zevallos, and Mustafa ES Akbulut. 2024. Integrated pest management: an update on the sustainability approach to crop protection. *ACS omega*, 9(40):41130–41147.

Yutong Zhou and Masahiro Ryo. 2024. Agribench: A hierarchical agriculture benchmark for multimodal large language models. In *European Conference on Computer Vision*, pages 207–223. Springer.

A Background Definitions

A.1 Definition of IPM Competencies

I define in Table 3 the core competencies required to solve workflow-oriented, knowledge-intensive integrated pest management tasks. Following the canonical IPM loop, IPM-Bench is organized into four stages: pest identification, knowledge retrieval, infestation assessment and reasoning, and control strategy recommendation. Each stage corresponds to distinct knowledge and skill requirements, ranging from entity-level recognition and differential diagnosis, to threshold-based decision making and integrated multi-tactic planning. These competencies provide an interpretable mapping from benchmark performance to practical IPM capabilities.

A.2 Example Definitions of IPM Terminology

I define in Table 4 specialized IPM terms used throughout this paper, providing a unified reference for task design, taxonomy interpretation, and evaluation of model outputs.

Stage	Corresponding Task	Competency	Description
Pest Identification	Pest Named Entity Recognition	Ability to recognize pest-related entities	Extracts pest names, host crops, symptoms, and life stages from short descriptions or extension-style text with correct normalization.
	Crop–Pest / Symptom–Pest Matching	Ability to link observations to likely pests	Maps observed symptoms and crop context to plausible pest causes using diagnostic cues, avoiding unsupported matches.
	Pest Identification, Single Choice	Ability to identify a pest from evidence	Selects the most likely pest given a description and context, reflecting common IPM identification queries.
	Pest Identification, Multi Answer	Ability to perform differential diagnosis	Produces a short set of plausible candidates when evidence is ambiguous, prioritizing likely causes and excluding implausible confounders.
Knowledge Retrieval	Cloze Facts on Pest Biology	Ability to recall precise IPM facts	Recovers exact factual spans such as host range, life-cycle attributes, damage patterns, or decision cues with correct units and wording.
	Open Factual Generation	Ability to generate concise grounded facts	Provides short factual answers grounded in the given evidence, emphasizing correctness over generic advice and minimizing hallucinated details.
Infestation Assessment and Reasoning	Threshold Decision, Single Choice	Ability to make threshold-based decisions	Determines whether monitoring observations exceed action or economic thresholds, reflecting the monitor–threshold–act logic of IPM.
	Scenario Reasoning, Multi Answer	Ability to reason over contextual factors	Identifies multiple correct factors or steps in a monitoring scenario (e.g., growth stage, sampling signals, environment, natural enemies) that affect assessment.
	Numeric Threshold Cloze	Ability to retrieve numeric decision rules	Extracts threshold values and units exactly (counts, percentages, per-leaf/per-trap metrics), which are critical for consistent decisions.
	Open-ended Reasoning	Ability to justify assessment decisions	Produces a concise rationale tying evidence to conclusions, explicitly referencing relevant cues (symptoms, counts, timing, stage).
	Action Classification	Ability to map assessment to action type	Classifies the appropriate next step (e.g., continue monitoring, apply control, adjust prevention), aligning decisions with IPM-compliant responses.
Control Strategy Recommendation	IPM Plan Generation	Ability to propose an integrated IPM plan	Generates a short multi-tactic plan consistent with extension guidance, prioritizing prevention and low-risk tactics and incorporating monitoring and timing.
	Key-information Extraction From Plans	Ability to extract actionable plan elements	Extracts key steps, timing, monitoring triggers, and constraints from an IPM plan to support execution and downstream decision making.

Table 3: Core IPM Competencies. IPM-Bench organizes integrated pest management into four workflow stages and 13 expert-curated task categories. Each task targets a specific competency needed for practical, extension-aligned IPM decision support.

Terminology	Definition
Integrated Pest Management (IPM)	A decision-making framework that combines monitoring, threshold-based decisions, and complementary tactics (cultural, biological, chemical, mechanical) to manage pests while minimizing risks to people and the environment.
Monitoring / Scouting	Field observation procedures used to detect pests, symptoms, and damage and to estimate pest pressure (e.g., trap counts, plant inspections, percent incidence) over time.
Action Threshold	A predefined pest density or damage level at which management action is recommended to prevent unacceptable injury; often expressed as a numeric rule tied to sampling units (e.g., pests per leaf, per trap, per plant).
Economic Threshold	The pest level at which expected economic loss equals the cost of control; crossing this threshold implies that intervention is economically justified.
Host Plant	A crop or plant species that a pest can feed on, reproduce on, or otherwise exploit; host information is central to diagnosis and management recommendations.
Life Stage	A developmental stage of an insect or pathogen (e.g., egg, larva, adult; lesion stage) that affects symptom expression, monitoring strategy, and control timing.
Natural Enemy	A beneficial organism that suppresses pests through predation, parasitism, or competition; natural-enemy presence can influence assessment and tactic choice.
Cultural Control	Non-chemical actions that reduce pest pressure through farming practices (e.g., rotation, sanitation, resistant varieties, planting date, canopy management).
Biological Control	Pest suppression achieved through natural enemies or biological products; may include conservation, augmentation, or classical biological control.
Chemical Control	Use of pesticides to manage pests; in IPM this is typically used judiciously and integrated with monitoring, thresholds, and resistance management.
Differential Diagnosis	A diagnostic process that enumerates multiple plausible causal agents for observed symptoms and narrows them using context (host, timing, distribution, and signs).

Table 4: IPM Terminology Definitions. Key concepts referenced throughout IPM-Bench task design, evaluation, and analysis.

B Source Data Collection

B.1 Construction and Annotation of High-Quality Text Segments

Each text segment extracted from the extension documents is annotated with four key fields that enable systematic question generation and evaluation. The **Content** field contains a snippet extracted by IPM domain experts from authoritative extension publications. The **Example Question** field provides an illustrative question demonstrating how one might inquire about the snippet, serving as a template for generating similar questions. The **Classification** field indicates the task type or IPM subfield to which the snippet belongs, ensuring proper categorization across the 13 task types. The **Reference** field identifies the source publication or extension program from which the snippet originates, maintaining traceability and credibility.

Figure 5 shows two representative annotated text segments from IPM-Bench. Each segment is stored as a structured record with four fields: **Content** (expert-selected snippet), **Example Question** (an illustrative query template), **Classification** (mapping to one of the 13 task types), and **Reference** (source provenance for traceability).

Content: Crop rotation has the ability to reduce the weeds, diseases, insects, and nematodes that harm alfalfa. A minimum rotation period of three years is recommended to effectively reduce nematode populations and prevent the buildup of pest populations that can significantly impact alfalfa yields.

Example Question: What is the minimum recommended rotation period for alfalfa to effectively reduce nematode populations?

Classification: Knowledge Retrieval (Single Choice)

Reference: UC Statewide IPM Program – Alfalfa Pest Management Guidelines

Content: The recommended practice is to hang three volatile-baited sphere traps in a 10- to 15-acre orchard, on the outside row facing the most probable direction of apple maggot migration (towards woods or abandoned apple trees, or else on the south-facing side). Then, the traps are periodically checked to get a total number of flies caught; dividing this by 3 gives the average catch per trap, and a spray is advised when the result is 5 or more. Research done in Geneva over a number of years indicates that these traps work so well that it is possible to use a higher threshold than the old "1 fly and spray" guidelines recommended for the panel traps.

Example Question: According to the text, what is the recommended threshold for spraying insecticides when using volatile-baited sphere traps to monitor apple maggot populations?

Classification: Infestation Assessment (Reading Comprehension)

Reference: Cornell University IPM Program – Apple Maggot Fact Sheet

Figure 5: Examples of high-quality text segments and their annotations in IPM-Bench. Each segment is curated from extension publications and labeled with four fields (**Content**, **Example Question**, **Classification**, **Reference**) to support systematic question generation, task categorization across the 13 task types, and source-traceable evaluation.

B.2 Prompt Templates

As shown in Figure 2, I use a unified prompt template to query Gemini-2.5-Pro once per text segment and generate all 13 task instances in a single pass. For readability and consistency, the task instructions are grouped into four IPM-aligned blocks: (1) Pest Identification (NER, crop–pest matching, and subtask classification), (2) Knowledge Retrieval Q&A (four QA formats), (3) Contextual Reading Comprehension (four RC formats), and (4) Summarization and Structured Extraction (simple summary and key-information extraction). The template also enforces JSON Lines output (exactly 13 JSON objects, one per line) and requires answers to be grounded in the input segment to reduce hallucinations and preserve traceability.

Figure 6: Prompt excerpts and representative JSON Lines outputs for the 13 task types. The original task instructions in the unified Gemini prompt are shown verbatim (blue) and organized here into four blocks for readability, while the generator still produces exactly 13 JSON objects per input segment in a single pass. Orange boxes show example JSONL outputs consistent with the required schemas.

(a) Knowledge Retrieval (QA) tasks

```
"task_type": "qa_single_choice": A multiple-choice question with one correct answer. The
↪  JSON must include: "question", a list of 4 strings called "choices", and an integer
↪  "answer" representing the 0-indexed correct choice.

"task_type": "qa_multi_choice": A multiple-choice question with one or more correct answers.
↪  The JSON must include: "question", a list of 4 strings called "choices", and a list of
↪  integers called "answers" representing the 0-indexed correct choices.

"task_type": "qa_fill_blank": A sentence with a key term replaced by _______. The JSON must
↪  include: "question" (the sentence with the blank) and "answer" (the string that fills
↪  the blank).

"task_type": "qa_generation": An open-ended question requiring a detailed answer. The JSON
↪  must include: "question" and a string "answer".
```

```
{"task_type":"qa_single_choice","question":"What is the minimum recommended rotation
↪  period?","choices":["1 year","2 years","3 years","4 years"],"answer":2}
{"task_type":"qa_fill_blank","question":"A minimum rotation period of _______ years is
↪  recommended.","answer":"three"}
```

(b) Summarization and key-information extraction tasks

```
"task_type": "sum_simple": A task to summarize the text. The JSON must include: "context"
↪  (the full source text), "question" (e.g., "Provide a brief summary."), and "answer" (a
↪  concise one or two-sentence summary).

"task_type": "sum_key_info": A task to extract key information. The JSON must include:
↪  "context", "question", and an "answer" which is a JSON object of key-value pairs.
```

```
{"task_type":"sum_simple","context":"...","question":"Provide a brief
↪  summary.","answer":"..."}
{"task_type":"sum_key_info","context":"...","question":"Extract key
↪  information.","answer":{"minimum_rotation":"3 years","target":"nematodes"}}
```

(c) Context-required reading comprehension tasks

```
"task_type": "rc_single_choice": A single-choice question that requires the provided context
↪  to answer. The JSON must include: "context", "question", "choices", and "answer".

"task_type": "rc_multi_choice": A multi-choice question that requires the provided context
↪  to answer. The JSON must include: "context", "question", "choices", and "answers".

"task_type": "rc_fill_blank": A fill-in-the-blank question that requires the provided
↪  context to answer. The JSON must include: "context", "question", and "answer".

"task_type": "rc_generation": An open-ended question that requires the provided context to
↪  answer. The JSON must include: "context", "question", and "answer".

"task_type": "rc_subtask_classification": A task to classify the text's purpose. The JSON
↪  must include: "context", "question", 4 choices in a "choices" list, and the correct
↪  0-indexed "answer".
```

```
{"task_type":"rc_single_choice","context":"Spray is advised when the average catch per trap
↪  is 5 or more.","question":"When is a spray
↪  advised?","choices":["1+","3+","5+","10+"],"answer":2}
```

(d) Matching and named-entity extraction tasks

```
"task_type": "crop_pest_matching": A question matching a pest/disease to its host crop. The
↪  JSON must include: "context", "question", "choices", and "answer".

"task_type": "agricultural_ner": A Named Entity Recognition task. The JSON must include:
↪  "text" (a sentence from the source) and "entities", a list of JSON objects. Each entity
↪  object must have "text", "type", "start", and "end". Use these entity types: PEST,
↪  DISEASE, CROP, SYMPTOM, TREATMENT, BIOCONTROL_AGENT, ACTION, LIFE_STAGE, PHYSICAL_CHAR.
```

```
{"task_type":"agricultural_ner","text":"Aphids infest lettuce
↪  seedlings.","entities":[{"text":"Aphids","type":"PEST","start":0,"end":6},{"text":"let
↪  tuce","type":"CROP","start":15,"end":22}]}
```

C Quality Verification

C.1 LLM as judge

From the 6,943 high-quality sentences, we used Gemini to synthesize candidates across 13 task types, producing approximately 90,259 QA items (one per task type per sentence). To reduce noise while preserving a sufficiently large candidate pool, we applied an LLM-as-judge stage: a separate evaluator LLM scored each item for answerability, faithfulness to the cited evidence, format adherence, and safety/IPM compliance. For each task type, we retained the top-300 highest-scoring items (300×13 = 3,900) for downstream verification. We then ran automatic validation checks (e.g., exact-match recoverability, overlap consistency, schema parsing, and near-duplicate removal), resulting in 3,158 validated candidates. Finally, two IPM experts reviewed the validated pool and selected/refined 200 items per task type, yielding the final IPM-Bench benchmark of 2,600 items.

Table 5 summarizes the reduction at each stage from initial generation to the final benchmark.

Stage	Count	Description
Input Sentences	6,943	High-quality sentences from extension documents
Generated Candidates	90,259	Gemini generation (13 task types × 6,943 sentences)
After LLM-as-Judge Filtering	3,900	Top-300 items per task type (300 × 13)
After Automatic Validation	3,158	Passed recoverability, format parsing, and dedup checks
Final IPM-Bench	2,600	Expert-selected/refined top-200 per task type (200 × 13)

Table 5: Quality verification pipeline for IPM-Bench. The LLM-as-judge stage reduces ∼91K generated candidates to a manageable pool (3,900) while preserving diversity; automatic validation removes technically invalid or non-recoverable items (3,158); domain experts then select and refine a balanced final set of 2,600 items (200 per task type).

Table 6 lists the rubric dimensions used by the judge model to score each candidate and rank candidates within each task type. Each item receives four scores on a 0 to 5 scale, covering answerability, faithfulness to the cited evidence, format adherence to the required JSON schema, and safety with respect to IPM compliant guidance. We aggregate these scores into an overall rating and retain the top 300 candidates per task type for the subsequent automatic validation stage.

Criterion	Description
Answerability	The question has a clear, unambiguous answer given the provided evidence/context.
Faithfulness	The reference answer is directly supported by the cited evidence (no hallucination).
Format Adherence	The output strictly matches the required JSON schema for the task type.
Safety / IPM Compliance	No unsafe pesticide instructions or non-IPM-compliant operational guidance.

Table 6: LLM-as-judge rubric used to score candidate items (0–5 per criterion) and rank them within each task type. We retain the top-300 items per task type for automatic validation.

To illustrate the LLM as a judge filtering process, Table 7 provides criterion specific contrasts between high scoring retained items and low scoring rejected items, making common failure patterns explicit. Figure 7 then presents the full judging protocol, including the rubric, the required inputs, and the machine readable JSON output schema, together with representative retained and rejected examples and their expected judge outputs.

Judge protocol (rubric + output schema).

Input: `{task_type, evidence, candidate_item}`

Rubric (0–5):

- **Answerability**: The question has a clear, unambiguous answer given the evidence/context.
- **Faithfulness**: The reference answer is directly supported by the cited evidence; no hallucination.
- **Format**: The output strictly matches the required JSON schema for the task type.
- **Safety**: No unsafe or non-IPM-compliant operational guidance.

Output (JSON):

```
{"answerability_score":0--5, "faithfulness_score":0--5,
 "format_score":0--5, "safety_score":0--5, "overall":0--5,
 "decision":"retain|reject", "rationale":"one sentence"}
```

High-score (retained).
Evidence (abridged):

"A minimum rotation period of three years is recommended to reduce nematode populations. . . "

Candidate item (abridged):

```
{"task_type":"qa_fill_blank",
 "question":"A minimum rotation period of ____ years is recommended...",
 "answer":"three"}
```

Expected judge output:

```
{"answerability_score":5,"faithfulness_score":5,"format_score":5,"safety_score":5,
 "overall":5.0,"decision":"retain",
 "rationale":"Answer is explicitly supported by the evidence; schema valid; safe."}
```

Low-score (rejected).
Evidence (abridged):

"Monitor aphids regularly and record population trends. . . "

Candidate item (abridged):

```
{"task_type":"qa_fill_blank",
 "question":"Apply treatment when there are ____ aphids per leaf.",
 "answer":"10"}
```

Expected judge output:

```
{"answerability_score":2,"faithfulness_score":1,"format_score":5,"safety_score":4,
 "overall":3.0,"decision":"reject",
 "rationale":"The numeric threshold is not supported by the cited evidence (hallucinated)."}
```

Figure 7: LLM-as-judge protocol for filtering generated candidates. The judge takes as input a task type, an evidence snippet, and a candidate item, then scores the item on four rubric dimensions (0–5): answerability, faithfulness to the evidence, format adherence to the required JSON schema, and safety/IPM compliance. The judge outputs a machine-parsable JSON record containing the four criterion scores, an overall score, a retain/reject decision, and a one-sentence rationale. The lower panels show representative high-score (retained) and low-score (rejected) examples, with their evidence, candidate items, and expected judge outputs.

Criterion	Retained (High Score)	Rejected (Low Score)
Answerability	*Question:* "What is the minimum recommended rotation period for alfalfa to effectively reduce nematode populations?" *Context:* "A minimum rotation period of three years is recommended..." *Score:* High – Clear answer in context	*Question:* "What is the optimal planting depth for all crops?" *Context:* "Plant alfalfa 1/4 to 1/2 inch deep..." *Score:* Low – Question too broad; context only covers alfalfa
Faithfulness	*Question:* "Which pests are mentioned as seedling pests?" *Answer:* ["Aphids", "Cutworms", "Flea beetles"] *Context:* "Watch for seedling pests. Aphids Crickets Cutworms Flea beetles..." *Score:* High – Answer directly from context	*Question:* "What is the economic threshold for aphids?" *Answer:* "5 aphids per leaf" *Context:* "Monitor aphids regularly. Apply treatment if needed." *Score:* Low – Answer not in context (hallucinated)
Format Adherence	`{"task_type": "qa_single_choice", "question": "...", "choices": ["A", "B", "C", "D"], "answer": 2}` *Score:* High – Valid JSON with all required fields	`{"task_type": "qa_single_choice", "question": "...", "options": ["A", "B"]}` *Score:* Low – Missing "choices" field; wrong field name; missing "answer"
Safety	*Question:* "What monitoring methods are recommended before applying pesticides?" *Answer:* "Use pheromone traps and scouting to determine pest levels before considering treatment." *Score:* High – Emphasizes monitoring first	*Question:* "What pesticide should be applied immediately?" *Answer:* "Apply broad-spectrum insecticide at maximum rate without monitoring." *Score:* Low – Non-IPM compliant; no monitoring or thresholds mentioned

Table 7: Concrete examples of LLM-as-judge evaluation: retained items (high scores) versus rejected items (low scores) for each evaluation criterion.

C.2 Two-round validation

We then performed a two-round validation combining automatic checks and domain-expert review. Figure 8 illustrates the complete validation workflow.

Round 1: Automatic Validation

- **Exact Match Verification:** Re-execute answer extraction for cloze and choice items to verify the reference answer can be recovered from the source passage.
- **Overlap Checks:** Apply textual or semantic overlap validation for short-answer items to ensure answerability.
- **Near Duplicate Detection:** Re-run duplicate checks at the item level to remove templated variations that differ only superficially.
- **Rejection:** Remove any item that fails answer recoverability, violates the required JSON schema, or is flagged as a near duplicate; this stage reduces the pool from 3,900 to 3,158.

Round 2: Expert Review

- **Independent Audit:** Two IPM experts independently review each item for:
  - Domain correctness (factual accuracy)
  - Clarity (question wording and comprehension)
  - Realism (practical relevance to IPM workflows)
- **Refinement Tasks:**
  - Edit question wording (normalize local crop names, improve clarity)
  - Reconcile label sets for entities and tactics
  - Flag non IPM compliant content (for example, recommending broad-spectrum chemical control without monitoring)
- **Selection:** From the 3,158 automatically validated items, experts select and refine the top 200 per task type, yielding the final 2,600 items.
- **Conflict Resolution:** Disagreements are adjudicated by consulting the original extension guidelines.

Output: Final IPM-Bench (2,600 items), curated from 3,158 automatically validated candidates and balanced to 200 items per task type across four IPM stages.

Figure 8: Two-round validation pipeline combining automatic checks and expert review. Candidates are first screened by an LLM as a judge (top 300 per task type, 3,900 total). Round 1 applies automatic validation for answer recoverability, strict schema compliance, and near duplicate removal, reducing the pool to 3,158. Round 2 performs expert auditing and refinement, then selects 200 items per task type to form the final IPM-Bench benchmark of 2,600 items.

C.2.1 Automatic Validation

First, automatic validators re-executed answer extraction to verify exact match for cloze/choice items and textual/semantic overlap for short-answer items (rejecting any question whose reference answer could not be recovered from its source passage). We also re-ran near-duplicate checks at the item level to remove templated variations that differed only superficially.

Table 8 illustrates the automatic validation process with concrete examples of items that passed versus failed validation checks.

C.2.2 Expert Review

Second, two IPM experts independently audited each retained item for domain correctness, clarity, and realism. They edited question wording (e.g., normalizing local crop names), reconciled label sets for entities and tactics, and flagged any item that suggested non-IPM-compliant behavior (e.g., recommending broad-spectrum chemical control without monitoring or thresholds). Disagreements were adjudicated with reference to extension guidelines from the original sources.

Validation Type	Passed	Failed (Rejected)
Exact Match	*Question:* "Plant alfalfa seeds at a depth of ____." *Answer:* "1/4 to 1/2 inch" *Context:* "Plant 1/4 to 1/2 inch deep, depending on soil type." *Status:* ✓ Exact match found	*Question:* "The recommended planting depth is ____." *Answer:* "2 inches" *Context:* "Plant 1/4 to 1/2 inch deep..." *Status:* × Answer not in context
Overlap Check	*Question:* "Describe the benefits of crop rotation." *Answer:* "Crop rotation reduces weeds, diseases, and nematodes." *Context:* "Crop rotation has the ability to reduce the weeds, diseases, insects, and nematodes..." *Status:* High semantic overlap (0.85)	*Question:* "What is the best planting time?" *Answer:* "Early spring is optimal for all regions." *Context:* "Central Valley: early fall (September through October)..." *Status:* Low overlap, contradicts context
Duplicate Detection	*Item 1:* "What is the minimum rotation period for alfalfa?" *Item 2:* "What rotation period is recommended for alfalfa?" *Status:* Retained – different phrasing	*Item 1:* "What is the minimum rotation period for alfalfa?" *Item 2:* "What is the minimum rotation period for alfalfa?" *Status:* Removed – exact duplicate

Table 8: Examples of automatic validation checks: items that passed (retained) versus failed (rejected).

Table 9 shows concrete examples of expert review modifications, demonstrating how items were refined for clarity, correctness, and IPM compliance.

The expert pass produced the final IPM-Bench set: a balanced collection aligned to the four IPM stages, with strict formats for automatic scoring and rich, practice-grounded content suitable for probing LLMs' identification, retrieval, reasoning, and recommendation abilities.

Review Aspect	Before Expert Review	After Expert Review
Clarity	*Question:* "What should you do during this time?" *Issue:* Vague, lacks context	*Question:* "What should you do during alfalfa stand establishment?" *Improvement:* Added specific context
Terminology	*Question:* "What is the common name for lucerne?" *Issue:* Uses British term "lucerne"	*Question:* "What is the common name for alfalfa (Medicago sativa)?" *Improvement:* Normalized to "alfalfa" with scientific name
IPM Compliance	*Question:* "When should you spray pesticides?" *Answer:* "Apply immediately when pests are observed." *Issue:* No mention of thresholds or monitoring	*Question:* "When should you consider applying pesticides for aphid control?" *Answer:* "Apply when aphid populations exceed economic thresholds (typically 40–50 per stem) and natural enemies are insufficient." *Improvement:* Emphasizes thresholds and biological controls
Entity Labels	*NER Output:* {"text": "bug", "type": "PEST"} *Issue:* Too generic, should be specific	*NER Output:* {"text": "alfalfa weevil", "type": "PEST"} *Improvement:* More specific pest identification
Realism	*Question:* "What is the best IPM strategy for all crops?" *Issue:* Overly general, not realistic	*Question:* "What IPM strategies are recommended for managing aphids in alfalfa during stand establishment?" *Improvement:* Context-specific, realistic scenario

Table 9: Examples of expert review modifications: before and after comparisons showing how items were refined for clarity, terminology normalization, IPM compliance, entity label specificity, and realism.

D Additional Experimental Setup

D.1 Model Hyperparameters

We conduct evaluations on different large language models using a local Hugging Face inference pipeline as the primary tool. In order to ensure reproducibility and provide a reference for future research, the main hyperparameters used in our experiments, along with their meanings, are listed in Table 10.

Table 10: Main Hyperparameters for LLM Evaluation

Parameter	Meaning	Value
`max_seq_len`	The maximum context length (upper limit of input tokens)	2048
`max_out_len`	The maximum output length (upper limit of generated tokens)	1024
`batch_size`	The batch size (number of requests processed per generation)	8

For our open-source LLMs, the maximum input length was truncated to 2048 tokens, and the maximum generation tokens were set to 1024, with a batch size of 8. We used the default decoding configuration in Hugging Face Transformers (`do_sample` = `False`), with default sampling hyperparameters (temperature = 1.0, top-p = 1.0, and top-k = 50).

D.2 Evaluation Metrics

Our evaluation consists of two stages: *(i) response standardization* and *(ii) task-specific scoring*. In Stage (i), we normalize raw model outputs into task-specific canonical forms (e.g., extracting option indices for multiple-choice questions, parsing comma-separated indices for multiple-answer questions, and trimming whitespace/punctuation for free-form generation). In Stage (ii), we compute metrics tailored to each task type:

Multiple Choice Tasks: Accuracy. For single-answer multiple-choice tasks, each example has a gold label $y_i \in \mathbb{N}$ and a predicted label $\hat{y}_i \in \mathbb{N}$ after standardization. Accuracy is defined as

$$\text{Acc} \;=\; \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\hat{y}_i = y_i\right], \tag{1}$$

where N is the number of evaluated examples and $\mathbb{I}[\cdot]$ is the indicator function.

Multiple Answer Tasks: Macro-F1. For tasks where multiple options may be correct, each example i has a gold set $Y_i \subseteq \mathbb{N}$ and a predicted set $\hat{Y}_i \subseteq \mathbb{N}$. We compute per-example precision and recall:

$$P_i \;=\; \frac{|\hat{Y}_i \cap Y_i|}{|\hat{Y}_i| + \epsilon}, \qquad R_i \;=\; \frac{|\hat{Y}_i \cap Y_i|}{|Y_i| + \epsilon}, \tag{2}$$

and the per-example F1:

$$F1_i \;=\; \frac{2P_iR_i}{P_i + R_i + \epsilon}. \tag{3}$$

We report Macro-F1 as the average across examples:

$$\text{Macro-F1} \;=\; \frac{1}{N}\sum_{i=1}^{N} F1_i, \tag{4}$$

where ϵ is a small constant to avoid division by zero (we use $\epsilon = 10^{-12}$ in our implementation).

Fill-in-the-Blank: ROUGE-L F1. For fill-in-the-blank tasks, we measure overlap between the prediction $\hat{s}$ and reference s using ROUGE-L based on the length of the longest common subsequence (LCS). Let $L = \mathrm{LCS}(\hat{s}, s)$, and let $|\hat{s}|$ and $|s|$ denote sequence lengths (at the token level used by ROUGE). ROUGE-L precision and recall are

$$P_{\mathrm{LCS}} = \frac{L}{|\hat{s}| + \epsilon}, \qquad R_{\mathrm{LCS}} = \frac{L}{|s| + \epsilon}. \tag{5}$$

ROUGE-L F1 is then

$$\text{ROUGE-L}_{F1} = \frac{2P_{\mathrm{LCS}}R_{\mathrm{LCS}}}{P_{\mathrm{LCS}} + R_{\mathrm{LCS}} + \epsilon}. \tag{6}$$

Generation Tasks: ROUGE-L F1 and BERTScore. For free-form generation (e.g., generative QA, reading comprehension, summarization), we report ROUGE-L F1 as the primary metric using the definition above. Additionally, we include BERTScore to capture semantic similarity beyond surface overlap. Given contextual embeddings from a pretrained encoder, let $\mathbf{h}_1, \ldots, \mathbf{h}_m$ be token embeddings for the prediction and $\mathbf{r}_1, \ldots, \mathbf{r}_n$ for the reference. Using cosine similarity $\cos(\cdot, \cdot)$, BERTScore precision and recall are

$$P_{\mathrm{BERT}} = \frac{1}{m}\sum_{j=1}^{m} \max_{k \in \{1,\ldots,n\}} \cos(\mathbf{h}_j, \mathbf{r}_k), \qquad R_{\mathrm{BERT}} = \frac{1}{n}\sum_{k=1}^{n} \max_{j \in \{1,\ldots,m\}} \cos(\mathbf{h}_j, \mathbf{r}_k), \tag{7}$$

and the BERTScore F1 is

$$F1_{\mathrm{BERT}} = \frac{2P_{\mathrm{BERT}}R_{\mathrm{BERT}}}{P_{\mathrm{BERT}} + R_{\mathrm{BERT}} + \epsilon}. \tag{8}$$

E Error Analysis of Large Language Model Outputs

Overview. We analyze model failures across multiple IPM-oriented tasks (e.g., single-choice QA, crop–pest matching, key-information extraction, and agricultural NER). Without computing explicit metrics, we summarize recurring error patterns observed in the outputs:

(i) *Output-format noncompliance* that breaks strict parsers.
(ii) *Wrong-option selection* caused by shallow lexical cues or incomplete reasoning.
(iii) *Schema drift* in structured (JSON) outputs where keys/structure deviate from the reference.
(iv) *Entity omission* (under-extraction) in information extraction tasks.
(v) *Span boundary/granularity errors* that fragment or over-extend entities.
(vi) *Label/role confusion* among semantically close categories (e.g., ACTION vs. TREATMENT).
(vii) *Spurious additions* (over-generation) that introduce extra entities not in the reference.

E.1 Strict Output-Format Violations

Problem Description. In several evaluation settings, the output format is *strict* (e.g., "output only the number of the correct option"). Models may still prepend/append explanatory text, punctuation, or option strings. Even when the underlying choice is correct, these format deviations can cause parsing failures or be treated as incorrect by automatic graders.

Example 1

Example Question:
What stage of almond fruit development corresponds to hull split, and when does it occur?
Reference:
1
Answer:
1. Mid-April to start of shaking
Analysis:
The model included the option text rather than outputting only the bare option number, violating the required format.

Example 2

Example Question:
What is the key activity performed after harvesting almonds?
Reference:
1
Answer:
1.
Analysis:
Although the intended option is recoverable, the output contains punctuation and does not match the strict "number-only" requirement.

Example 3

Example Question:
Match the crop with its most relevant pest or disease based on the context (Black cherry).
Reference:
2
Answer:
2. Cherry slug
Analysis:
The model appended the option string, which can break exact-format evaluation despite the correct index.

Figure 9: Examples of strict output-format violations in multiple-choice tasks (correct choice but invalid format).

E.2 Incorrect Option Selection and Semantic Confusion

Problem Description. The model sometimes selects an incorrect option due to lexical shortcuts (e.g., matching the pest name to the crop name), partial reading of the context, or confusion among similar symptoms/definitions. These errors reflect *semantic* failures rather than formatting issues.

Example 1

Example Question:
What crop is Peach Twig Borer a pest of?
Reference:
2 (Plum)
Answer:
1. Peach
Analysis:
The model likely relied on the substring "Peach" in the pest name and ignored the actual host association indicated by the reference.

Example 2

Example Question:
How large is an adult moth of the army cutworm?
Reference:
1 (About 1 inch long)
Answer:
2. About 1.5 inches long
Analysis:
The model chose a nearby but incorrect quantitative description, suggesting weak grounding in the provided choices.

Example 3

Example Question:
What is a common symptom of Fusarium Foot Rot?
Reference:
1 (Interveinal chlorosis and necrotic spotting)
Answer:
2. Wilting without spotting
Analysis:
The prediction reflects symptom-level confusion, selecting a plausible but wrong manifestation.

Figure 10: Examples of incorrect option selection driven by semantic confusion.

E.3 Schema Drift in Structured Outputs

Problem Description. For structured-generation tasks (e.g., "output a single JSON object"), the model may drift from the expected schema: it may use different key names than the reference, introduce extra categories, omit required fields, or wrap JSON in Markdown fences. Such deviations hurt downstream consumption even if the content is partially correct.

Example 1

Example Question:
Extract the most critical pieces of information for pest management as a single JSON object (Texas forage crops IPM overview).
Reference:
Keys like `Publication Topic`, `Target Crops`, `Geographic Focus`
Answer:
Keys like `Pests`, `Crops` (wrapped in "`json fences)
Analysis:
The model produced a valid-looking JSON but with a different key schema than the reference, causing evaluation and integration mismatch.

Example 2

Example Question:
Extract key pest-management information (bacterial crown rot passage) as one JSON object.
Reference:
`spread_mechanism`, `survival_environment`
Answer:
`Pests`, `Symptoms`, `Transmission`, `Prevention` (wrapped in "`json fences)
Analysis:
The model reorganized information into its own taxonomy instead of the reference schema, leading to key-level misalignment.

Example 3

Example Question:
Extract key pest-management information (root weevil/root beetle management) as one JSON object.
Reference:
`Cultural Controls`, `Organic Methods`, `Chemical Controls`, `Biological Controls`
Answer:
`Pests`, `Symptoms`, `Treatments`, plus extra categories
Analysis:
The model over-generalized into broad headings (e.g., `Treatments`) and deviated from the target control-method schema.

Figure 11: Examples of schema drift and key misalignment in JSON-style tasks.

E.4 Entity Omission and Under-Extraction

Problem Description. In extraction tasks (e.g., NER), the model may return an empty list or an incomplete set of entities even when clear mentions exist. This is common in longer sentences, enumerations, or when entity mentions are not prototypical (e.g., "natural enemies" as a BIOCONTROL_AGENT).

Example 1

Example Question:
Extract all entities from: "... least harmful to natural enemies, honey bees, and the environment ..."
Reference:
pesticides (TREATMENT), natural enemies (BIOCONTROL_AGENT), honey bees (BIOCONTROL_AGENT)
Answer:

Analysis:
The model failed to extract any entities from a sentence containing multiple salient IPM concepts.

Example 2

Example Question:
Extract all entities from: "Methomyl is also toxic to bees ..."
Reference:
Methomyl (TREATMENT)
Answer:

Analysis:
A critical chemical name was omitted, which would directly impact decision support and risk communication.

Example 3

Example Question:
Extract all entities from: "Apical stems are thick and assume an upright growth habit."
Reference:
upright growth habit (PHYSICAL_CHAR)
Answer:

Analysis:
The model missed a descriptive physical trait, indicating under-sensitivity to morphology/phenotype expressions.

Figure 12: Examples of entity omission (false negatives) in agricultural NER.

E.5 Span Boundary and Granularity Errors

Problem Description. Span boundary errors occur when the extracted text does not match the reference span: the model may fragment a single entity into multiple pieces, or over-extend into surrounding context. These errors reduce entity linking accuracy and can degrade evaluation even if the general meaning is correct.

Example 1

Example Question:
Extract entities from: "The pistachio seed chalcid overwinters as a diapausing larva in infested nuts."
Reference:
pistachio seed chalcid (PEST), diapausing larva (LIFE_STAGE), nuts (CROP)
Answer:
pistachio (CROP), seed chalcid (PEST), diapausing larva (LIFE_STAGE), infested nuts (PHYSICAL_CHAR)
Analysis:
The model split a single pest name into two entities and over-extended "nuts" into "infested nuts," creating boundary mismatches.

Example 2

Example Question:
Extract entities from: "Young worms are reddish orange . . . "
Reference:
Young worms (LIFE_STAGE), reddish orange (PHYSICAL_CHAR), cream-colored (PHYSICAL_CHAR)
Answer:
worms (PEST), Young (LIFE_STAGE), reddish orange (PHYSICAL_CHAR), cream-colored (PHYSICAL_CHAR)
Analysis:
The model fragmented a single life-stage span ("Young worms") into two parts and reassigned roles across them.

Example 3

Example Question:
Extract entities from: ". . . other tortricid caterpillars . . . "
Reference:
tortricid (PEST)
Answer:
tortricid caterpillars (PEST)
Analysis:
The model over-extended the annotated span by adding a head noun, producing a systematic boundary mismatch.

Figure 13: Examples of span boundary and granularity errors in agricultural NER.

E.6 Label and Role Confusion

Problem Description. Even with correct spans, the model may assign incorrect labels, especially among semantically close categories (e.g., ACTION vs. TREATMENT, BIOCONTROL_AGENT vs. TREATMENT, or DISEASE vs. PEST). This indicates ontology-level ambiguity and incomplete task grounding.

Example 1

Example Question:
Extract entities from: "The causal fungus, Verticillium dahliae, infects susceptible plants through the roots."
Reference:
Verticillium dahliae (DISEASE), roots (PHYSICAL_CHAR)
Answer:
Verticillium dahliae (PEST), roots (PHYSICAL_CHAR)
Analysis:
The model misclassified a causal pathogen/disease mention as a pest, reflecting entity-role confusion.

Example 2

Example Question:
Extract entities from: "Generally, dormant sprays . . . keep Italian pear scale under control . . . "
Reference:
dormant sprays (ACTION), Italian pear scale (PEST), moss (PHYSICAL_CHAR), lichens (PHYSICAL_CHAR)
Answer:
dormant sprays (TREATMENT), Italian pear scale (PEST), . . .
Analysis:
The span is recognized, but the label shifts from an application action to a treatment category, misaligning the ontology.

Example 3

Example Question:
Extract entities from: "The Bt program used for peach twig borer control reduces cankerworm damage."
Reference:
Bt program (TREATMENT), peach twig borer (PEST), cankerworm (PEST)
Answer:
Bt (BIOCONTROL_AGENT), peach (CROP), twig borer (PEST), cankerworm (PEST)
Analysis:
The model decomposed the treatment into a biocontrol agent mention and restructured the pest name, causing label and span misalignment.

Figure 14: Examples of label/role confusion among closely related categories.

E.7 Spurious Additions and Over-Generation

Problem Description. Spurious outputs arise when the model adds entities not present in the reference annotations (false positives), often by promoting contextual words (time, life stage markers, locations) into entities. This can happen due to over-eager extraction heuristics or weak calibration about what the benchmark considers "relevant."

Example 1

Example Question:
Extract entities from: "Cutworms are mainly active at night."
Reference:
Cutworms (PEST)
Answer:
Cutworms (PEST), night (PHYSICAL_CHAR)
Analysis:
The model promoted a time expression into an entity type that is not annotated in the reference, creating a false positive.

Example 2

Example Question:
Extract entities from: "The orange tortrix overwinters as larvae . . . in coastal areas."
Reference:
orange tortrix (PEST), larvae (LIFE_STAGE)
Answer:
orange tortrix (PEST), larvae (LIFE_STAGE), coastal areas (LOCATION)
Analysis:
A plausible location was extracted, but it is not included in the reference annotations for this instance.

Example 3

Example Question:
Extract entities from: "Adult consperse stink bugs have gray brown to green bodies . . . "
Reference:
consperse stink bugs (PEST), gray brown to green (PHYSICAL_CHAR), yellow to orange (PHYSICAL_CHAR)
Answer:
Adult (LIFE_STAGE), consperse stink bugs (PEST), . . .
Analysis:
The model introduced an extra life-stage entity ("Adult") that is not annotated in the reference, yielding an over-generation error.

Figure 15: Examples of spurious entities (false positives) in agricultural NER.

F Prompt Templates

Prompt for Task QA-1 (Single-Choice)

Topic Description:
Answer single-choice questions by selecting the best option index.

User:
You are an expert in Integrated Pest Management (IPM) for agriculture. Read the following question and choose the best answer from the options provided. Your answer should only be the number of the correct option.

Question: `{question}`

Options:
`0. {choice_0}`
`1. {choice_1}`
`2. {choice_2}`
`3. {choice_3}`

The number of the correct option is:

Example:
Question: Which practice best reduces the risk of pesticide resistance in insect pests?
Options:
0. Use the same insecticide mode of action every time for consistency.
1. Increase the dose above the label rate to ensure complete control.
2. Rotate insecticides with different modes of action across applications.
3. Apply insecticides only after visible crop damage becomes severe.

Answer: 2

Prompt for Task QA-2 (Multiple-Answer)

Topic Description:
Select all correct option indices and output them separated by commas.

User:
You are an expert in Integrated Pest Management (IPM) for agriculture. Read the following question and select ALL correct answers from the options provided. Multiple options may be correct. Your answer should be the numbers of all correct options, separated by commas.

Question: `{question}`

Options:
```
0. {choice_0}
1. {choice_1}
2. {choice_2}
3. {choice_3}
```

The numbers of the correct options are:

Example:
Question: Which of the following are common IPM tactics (select all that apply)?
Options:
0. Regular scouting and monitoring of pest populations.
1. Relying only on calendar-based spraying without thresholds.
2. Using biological control agents (e.g., parasitoids, predators).
3. Implementing crop rotation or sanitation to reduce pest pressure.

Answer: 0,2,3

Prompt for Task RC-1 (Single-Choice w/ Context)

Topic Description:
Answer single-choice questions grounded in the provided context passage.

User:
You are an expert in Integrated Pest Management (IPM) for agriculture. Read the following context and answer the multiple-choice question based on it. Your answer should only be the number of the correct option.

Context: `{context}`

Question: `{question}`

Options:
```
0. {choice_0}
1. {choice_1}
2. {choice_2}
3. {choice_3}
```

The number of the correct option is:

Example:
Context: Late summer nitrogen applications can increase Spring Dead Spot severity because added nitrogen promotes leaf growth at the expense of roots and can delay bermudagrass dormancy, reducing cold hardiness and carbohydrate reserves.
Question: According to the context, why can late summer nitrogen increase Spring Dead Spot severity?
Options:
0. It increases root development and improves dormancy timing.
1. It promotes leaf growth while reducing root reserves and delays dormancy.
2. It lowers soil pH, which directly kills the pathogen.
3. It increases cold hardiness by storing more carbohydrates.

Answer: 1

Prompt for Task RC-2 (Multiple-Answer w/ Context)

Topic Description:
Select all correct options grounded in the provided context passage.

User:
You are an expert in Integrated Pest Management (IPM) for agriculture. Read the following context and select ALL correct answers from the options provided. Multiple options may be correct. Your answer should be the numbers of all correct options, separated by commas.

Context: `{context}`

Question: `{question}`

Options:
`0. {choice_0}`
`1. {choice_1}`
`2. {choice_2}`
`3. {choice_3}`

The numbers of the correct options are:

Example:
Context: Maintaining a low soil pH (5.2–5.3) appears to reduce Spring Dead Spot severity. Using ammonium-based nitrogen sources is recommended over nitrate- or urea-based fertilizers.
Question: Which recommendations are supported by the context (select all that apply)?
Options:
0. Maintain soil pH around 5.2–5.3 to reduce disease severity.
1. Prefer ammonium-based nitrogen sources over nitrate- or urea-based fertilizers.
2. Apply high nitrogen in late summer to encourage rapid leaf growth.
3. Raise soil pH above 7.0 to suppress the disease.

Answer: 0,1

Prompt for Task QA-3 / RC-3 (Fill-in-the-Blank)

Topic Description:
Fill the blank denoted by `[BLANK]` with a word or phrase.

User:
You are an expert in Integrated Pest Management (IPM). Read the provided context and fill in the blank, which is represented by [BLANK], in the sentence that follows. Provide only the word or phrase that should go in the blank.

Context: `{context}`

Sentence: `{question_with_blank}`

Example:
Context: IPM emphasizes using action thresholds to decide when pest populations justify control measures.
Sentence: In IPM, an `[BLANK]` threshold helps determine when control actions are economically justified.

Answer: economic

Prompt for Task QA-4 / RC-4 (Generation)

Topic Description:
Provide a direct and concise free-form answer, grounded in context when available.

User:
You are an expert in Integrated Pest Management (IPM) for agriculture. Based on the context provided, please provide a direct and concise answer to the following question.

Context: `{context}`

Question: `{question}`

Example:
Context: Late summer nitrogen applications can delay bermudagrass dormancy and reduce cold hardiness and carbohydrate reserves, which makes plants more vulnerable when emerging in spring.
Question: Explain why minimizing nitrogen applications in late summer and fall helps manage Spring Dead Spot.

Answer: Late-season nitrogen promotes leaf growth while weakening roots and delaying dormancy, which reduces cold hardiness and carbohydrate reserves. This leaves bermudagrass less resilient through winter and more susceptible to Spring Dead Spot.

Prompt for Task SUM-1 (Simple Summarization)

Topic Description:
Write a brief one-paragraph summary focusing on IPM-relevant information.

User:
As an expert in Integrated Pest Management (IPM), provide a brief, one-paragraph summary of the following text, focusing on information relevant to pest management.

Text: `{context}`

Example:
Text: Scout fields weekly for aphids, especially during warm, dry periods. If populations exceed the action threshold, prioritize natural enemies and selective insecticides. Avoid unnecessary broad-spectrum sprays that can disrupt beneficial insects and increase resistance risk.

Summary: Monitor aphids regularly and apply control only when thresholds are exceeded. Favor biological control and selective insecticides to preserve beneficial insects, and avoid broad-spectrum sprays to reduce resistance and secondary pest outbreaks.

Prompt for Task SUM-2 / DS-3 (Key Information Extraction)

Topic Description:
Extract key IPM-relevant information into a single JSON object; output `{}` if nothing is found.

User:
As an expert in Integrated Pest Management (IPM), read the following text and extract the most critical pieces of information for pest management. Present them as a single JSON object, where keys represent categories of information (e.g., "Pests", "Symptoms", "Treatments") and values are a list of the extracted key phrases. For example: `{"Pests": ["aphids"], "Treatments": ["neem oil"]}`. If no key information is found, you must output an empty JSON object: `{}`.

Text: `{context}`

Example:
Text: Early blight on tomato often appears as dark concentric lesions on older leaves. Manage it by removing infected debris, using crop rotation, improving airflow, and applying labeled fungicides when conditions favor disease.

Output: `{"Pests": [], "Diseases": ["early blight"], "Symptoms": ["dark concentric lesions on older leaves"], "Treatments": ["remove infected debris", "crop rotation", "improve airflow", "apply labeled fungicides"]}`

Prompt for Task NER (Agricultural NER)

Topic Description:
Extract entities and output a JSON list of `{"text":..., "label":...}` objects; output `[]` if none.

User:
As an expert in Integrated Pest Management (IPM), your task is to extract all relevant entities from the text below. The possible entity types are: `ACTION,BIOCONTROL_AGENT,CROP,DISEASE,LIFE_STAGE,LOCATION, ORGANIZATION,PEST,PHYSICAL_CHAR,SYMPTOM,TREATMENT`. Format your output as a single JSON-formatted list of objects. Each object must have two keys: `text` (the extracted entity string) and `label` (the entity type). If no entities are found, you must output an empty list: `[]`.

Text: `{text}`

Example:
Text: Scout soybean fields for aphids. If aphid nymphs are abundant, consider releasing lady beetles and applying neem oil as a treatment.

Output:
`[{"text":"soybean","label":"CROP"}, {"text":"aphids","label":"PEST"}, {"text":"nymphs","label":"LIFE_STAGE"}, {"text":"scout","label":"ACTION"}, {"text":"lady beetles","label":"BIOCONTROL_AGENT"}, {"text":"neem oil","label":"TREATMENT"}]`

G Detailed Performance Comparison and Analysis

This appendix provides a comprehensive, fine-grained quantitative analysis of model performance on IPM-Bench, addressing the three research questions (RQ1–RQ3) introduced in Section 1. Building upon Table 2 (13 tasks $\times$ 24 models), we conduct diagnostic decomposition to reveal performance patterns, bottlenecks, and model-specific strengths across different task types and workflow stages.

G.1 Overall Comparison Across Model Families

We first examine performance differences across three model categories: *Proprietary* (GPT-4o, Gemini-2.0-Flash, etc.), *Open-source* (Qwen2.5-72B, Llama-3.1-70B, etc.), and *Domain-specific* (Aksara-v1, CropSeek, PLLaMa). Figure 16 shows the distribution of Average Relative Value (AVR) scores across these categories.

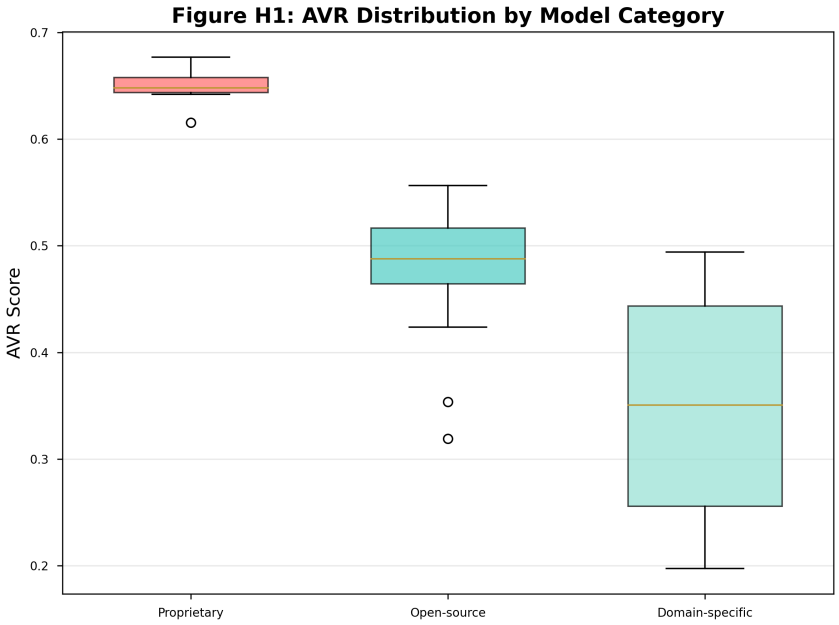

Figure 16: AVR Distribution by Model Category. The box plots show median, quartiles, and outliers for each category. Proprietary models demonstrate superior overall performance, while domain-specific models show high variance.

Table 11 presents group mean scores for each task type. Key observations:

- **Proprietary models** lead across most tasks, particularly in exact-span extraction (QA-3, RC-3) and generation tasks (QA-4, RC-4, SUM-1, SUM-2).
- **Open-source models** achieve competitive performance on constrained classification tasks (QA-1, RC-1, DS-1, DS-2) but struggle with format-sensitive cloze tasks.
- **Domain-specific models** show mixed results: while some (e.g., Aksara-v1) approach mid-tier open-source performance on identification tasks, they significantly lag in schema-sensitive and generation tasks.

The radar charts in Figure 17 visualize the performance profile of each individual model within each category across all 13 tasks. The first three charts show all models in Proprietary, Open-source, and Domain-specific categories respectively (each model in a different color), revealing both category-level patterns and within-category variance. The rightmost chart compares the category averages for a clear overview.

G.2 Performance by IPM Workflow Stages

Following the four-stage IPM workflow defined in Section 3.2, we map the 13 tasks to stages:

- **Pest Identification**: QA-1, QA-2, DS-2, DS-3
- **Knowledge Retrieval**: QA-3, QA-4

Table 11: Group Mean Scores by Task Type

Task	Proprietary	Open-source	Domain-specific
DS-1	0.994	0.970	0.761
DS-2	0.943	0.894	0.621
DS-3	0.525	0.298	0.234
QA-1	0.811	0.685	0.375
QA-2	0.815	0.704	0.728
QA-3	0.483	0.059	0.031
QA-4	0.240	0.164	0.092
RC-1	0.972	0.907	0.601
RC-2	0.909	0.705	0.716
RC-3	0.891	0.163	0.066
RC-4	0.472	0.309	0.180
SUM-1	0.282	0.173	0.094
SUM-2	0.109	0.090	0.027

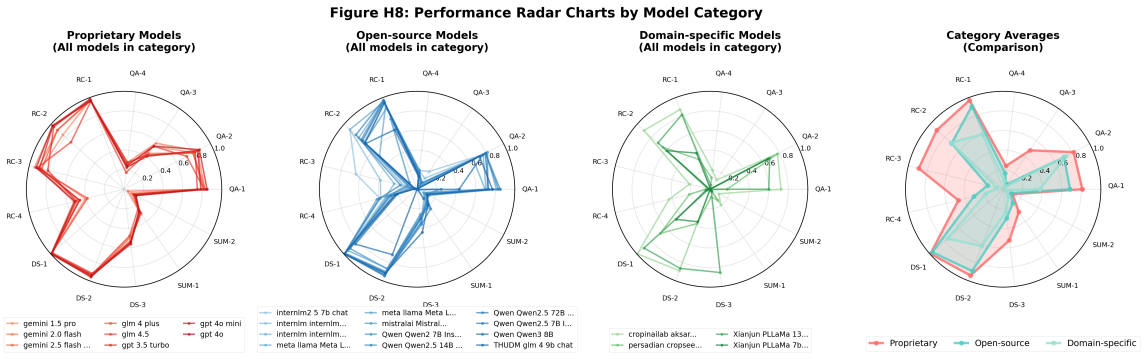

Figure 17: Performance Radar Charts by Model Category. Radar charts show individual model performance profiles within each category (Proprietary, Open-source, Domain-specific) across all 13 tasks in zero-shot settings. Each model is shown in a different color/shade within its category. The rightmost chart compares category averages.

- **Infestation Assessment & Reasoning**: RC-1, RC-2, RC-3, RC-4, DS-1
- **Control Strategy Recommendation**: SUM-1, SUM-2

Figure 18 visualizes each model's performance across these stages. The analysis reveals:

- **Bottleneck stages**: Control Strategy Recommendation (SUM-1, SUM-2) shows the lowest average scores across all models, indicating that evidence-grounded synthesis remains challenging.
- **Stage imbalance**: Many models (especially open-source) show strong performance in Pest Identification but significant drops in Knowledge Retrieval and Control Strategy Recommendation, supporting the "workflow-level gap" argument in Section 4.3.1.

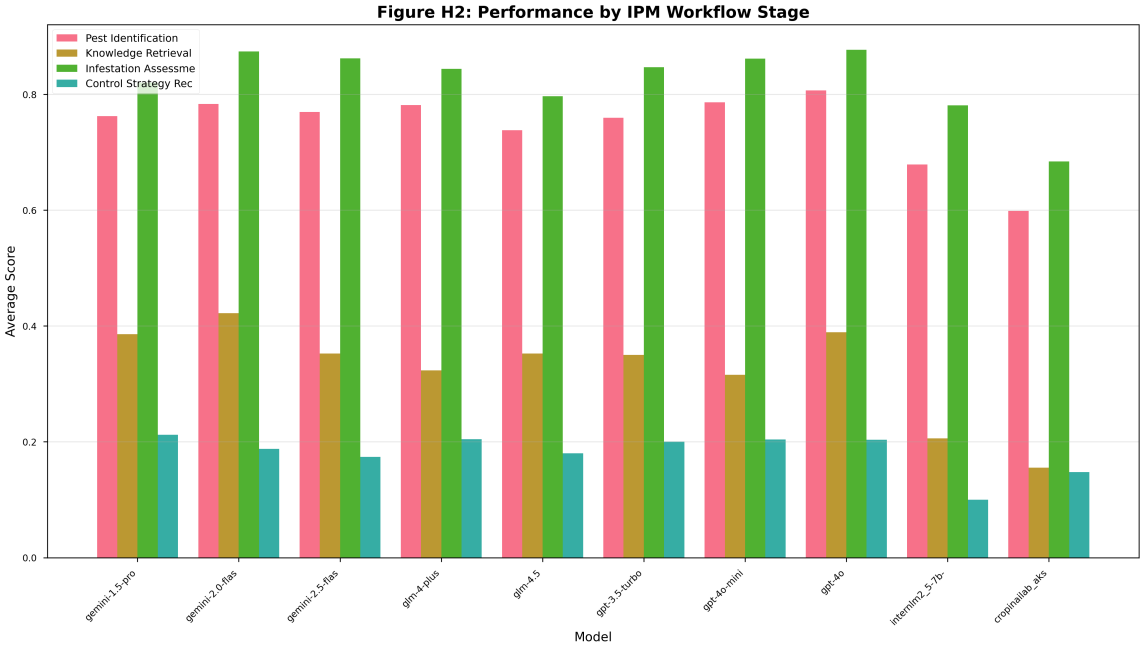

Figure 18: Performance by IPM Workflow Stage. Each bar group represents a model's average score across tasks in that stage. The "Control Strategy Recommendation" stage consistently shows the lowest scores.

Table 12 lists the top-3 models per stage. Notably, proprietary models dominate all stages except Knowledge Retrieval, where Gemini-2.0-Flash leads.

Table 12: Top-3 Models per Workflow Stage

Stage	Top-1	Top-2	Top-3
Pest Identification	gpt-4o (0.807)	gpt-4o-mini (0.786)	gemini-2.0-flash (0.783)
Knowledge Retrieval	gemini-2.0-flash (0.422)	gpt-4o (0.389)	gemini-1.5-pro (0.386)
Infestation Assessment & Reasoning	gpt-4o (0.877)	gemini-2.0-flash (0.874)	gemini-2.5-flash-lite (0.862)
Control Strategy Recommendation	gemini-1.5-pro (0.212)	glm-4-plus (0.205)	gpt-4o-mini (0.204)

G.3 Task Difficulty Spectrum & Bottleneck Tasks

To quantify task difficulty, we compute the average score across all models for each task. Figure 19 ranks tasks from easiest (highest average) to hardest (lowest average).

We further analyze models' "controllability" by grouping tasks into three categories:

- **Constrained classification**: QA-1, RC-1, DS-1, DS-2 (single/multi-choice, matching)
- **Exact-span / schema-sensitive cloze**: QA-3, RC-3 (fill-in-the-blank with strict format requirements)
- **Open-ended generation**: QA-4, RC-4, SUM-1, SUM-2 (free-form generation)

Figure 20 compares models' performance across these categories. The analysis reveals a critical finding: many open-source and domain-specific models achieve reasonable scores on constrained tasks but collapse

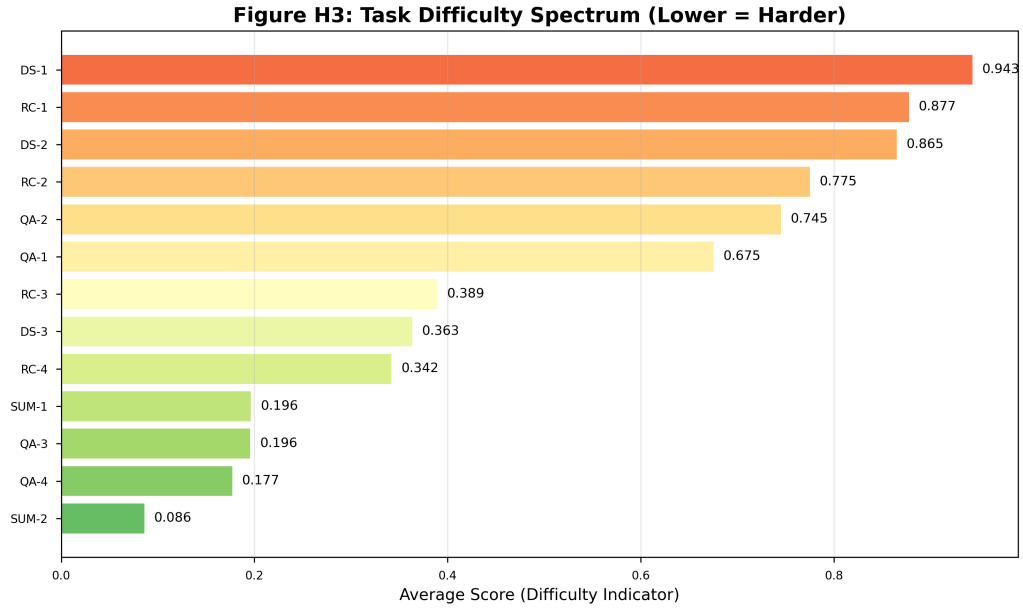

Figure 19: Task Difficulty Spectrum. Tasks are sorted by average score across all models (lower scores indicate higher difficulty). Fill-in-the-blank tasks (QA-3, RC-3) and summarization tasks (SUM-1, SUM-2) are the most challenging.

on cloze tasks, indicating that format sensitivity and exact extraction are major failure modes.

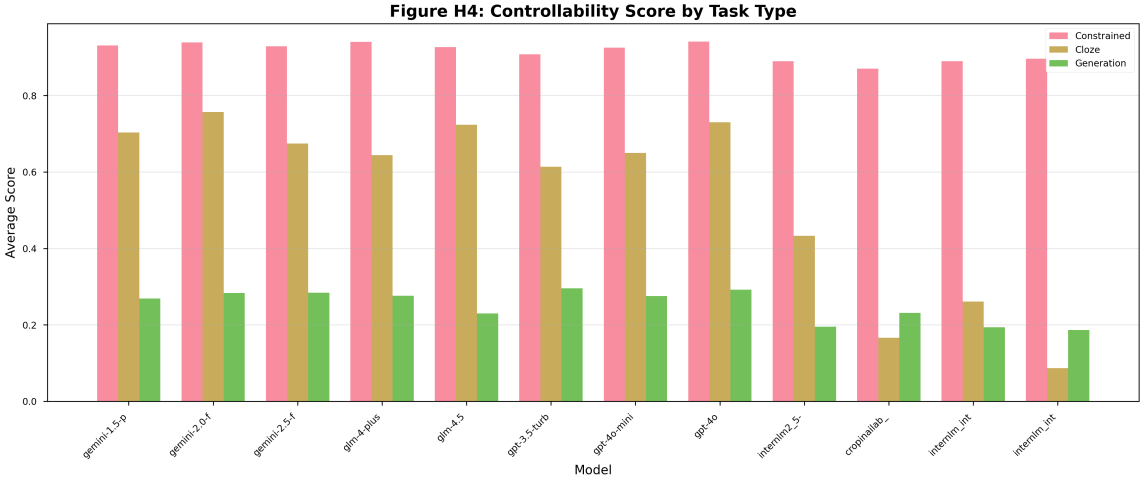

Figure 20: Controllability Score by Task Type. The gap between "Constrained" and "Cloze" scores (shown in Table 13) quantifies format sensitivity. Proprietary models show smaller gaps, indicating better controllability.

Table 13 reports the controllability gap (Constrained – Cloze) for each model. This metric directly supports the argument in Section 4.3.1 that many models fail not due to lack of knowledge, but due to inability to follow strict output formats.

G.4 Top Performers per Task Type

Table 15 lists the top-3 models for each task, enabling practitioners to select the best model for specific IPM sub-tasks. Table 15 provides stability statistics: the number of tasks where each model appears in the top-3 (or top-1).

Key insights:

- **Task-specific specialization**: Even the overall best models (GPT-4o, Gemini-2.0-Flash) do not dominate all tasks. For example, Qwen2.5-72B leads on QA-1, and PLLaMa-13B leads on DS-3 (Pest NER).

Table 13: Controllability Scores by Model

Model	Constrained	Cloze	Generation	Gap (C–C)
Qwen_Qwen2.5-14B-Instruct	0.934	0.014	0.174	0.920
THUDM_glm-4-9b-chat	0.906	0.017	0.170	0.889
Qwen_Qwen2.5-7B-Instruct	0.905	0.035	0.183	0.870
meta-llama_Meta-Llama-3.1-8B-Instruct	0.854	0.016	0.146	0.838
Qwen_Qwen2.5-72B-Instruct	0.941	0.124	0.230	0.817
internlm_internlm3-8b-instruct	0.896	0.087	0.187	0.809
mistralai_Mistral-7B-Instruct-v0.3	0.872	0.068	0.235	0.805
Xianjun_PLLaMa-13b-instruct	0.796	0.004	0.023	0.792
meta-llama_Meta-Llama-3.1-70B-Instruct	0.919	0.158	0.191	0.761
cropinailab_aksara_v1	0.870	0.166	0.231	0.704
Qwen_Qwen3-8B	0.660	0.014	0.116	0.646
internlm_internlm2_5-7b-chat	0.890	0.261	0.193	0.629
Qwen_Qwen2-7B-Instruct	0.700	0.103	0.189	0.597
internlm2_5-7b-chat	0.890	0.433	0.195	0.457
persadian_cropseek-llm	0.389	0.016	0.120	0.372
Xianjun_PLLaMa-7b-instruct	0.304	0.008	0.020	0.296
glm-4-plus	0.940	0.644	0.276	0.296
gpt-3.5-turbo	0.907	0.614	0.296	0.294
gpt-4o-mini	0.925	0.649	0.275	0.276
gemini-2.5-flash-lite	0.929	0.674	0.284	0.255
gemini-1.5-pro	0.931	0.704	0.269	0.228
gpt-4o	0.941	0.730	0.292	0.211
glm-4.5	0.926	0.724	0.230	0.203
gemini-2.0-flash	0.939	0.756	0.284	0.182

- **Stability ranking**: Proprietary models show the highest top-3 counts, confirming their robustness across diverse task types.

G.5 Model-Scale Analysis Within Major Families

To address RQ3 (scale vs. domain capability tradeoff), we examine scaling patterns within major open-source families. Figure 21 plots AVR against model size for Qwen, Llama, and InternLM families.

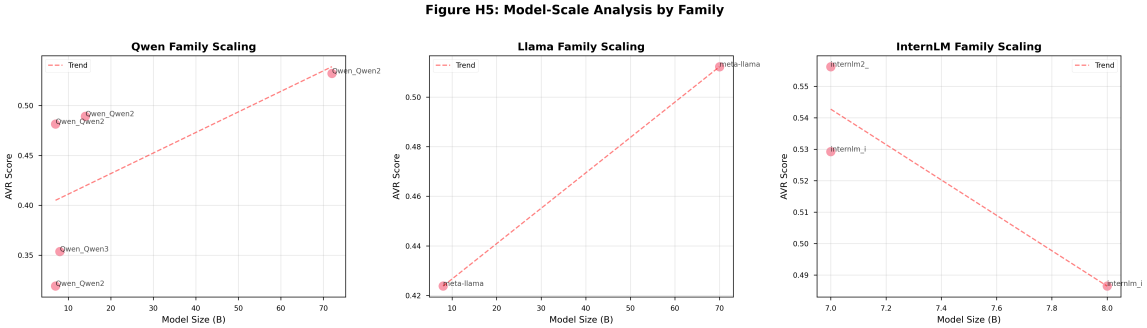

Figure 21: Model-Scale Analysis by Family. Each subplot shows AVR vs. model size for one family. Trend lines indicate positive scaling, but significant variance exists at the same scale (see Table 16).

Observations:

- **Positive scaling**: All families show increasing AVR with size, but the relationship is not linear.
- **Same-scale variance**: Table 16 shows that at similar scales (e.g., ∼7B), performance varies substantially. For instance, InternLM2.5-7B outperforms Llama-3.1-8B and Qwen2-7B, indicating that scale is not the sole determinant of performance (consistent with Section 4.3.3's "dispersion" argument).

Table 14: Top-3 Models per Task

Task	Top-1	Top-2	Top-3
DS-1	gemini-1.5-pro (1.000)	gemini-2.5-flash-lite (1.000)	gemini-2.0-flash (1.000)
DS-2	glm-4-plus (0.960)	gpt-4o (0.955)	gemini-2.0-flash (0.955)
DS-3	Xianjun_PLLaMa-13b-instruct (0.857)	gpt-4o-mini (0.565)	gpt-4o (0.551)
QA-1	Qwen_Qwen2.5-72B-Instruct (0.850)	gpt-4o (0.850)	gemini-2.0-flash (0.830)
QA-2	gpt-4o (0.870)	gpt-4o-mini (0.854)	gemini-2.0-flash (0.847)
QA-3	gemini-2.0-flash (0.570)	gpt-4o (0.536)	glm-4.5 (0.532)
QA-4	gpt-3.5-turbo (0.277)	gemini-2.0-flash (0.273)	gemini-2.5-flash-lite (0.261)
RC-1	gpt-4o-mini (0.985)	gpt-3.5-turbo (0.980)	gemini-1.5-pro (0.975)
RC-2	gpt-4o (0.980)	gpt-4o-mini (0.973)	gemini-2.0-flash (0.971)
RC-3	gemini-2.0-flash (0.943)	gpt-4o (0.925)	glm-4.5 (0.915)
RC-4	gemini-2.5-flash-lite (0.529)	gpt-4o (0.518)	gpt-3.5-turbo (0.507)
SUM-1	gemini-2.5-flash-lite (0.305)	glm-4-plus (0.296)	gemini-2.0-flash (0.291)
SUM-2	gpt-4o (0.140)	gemini-1.5-pro (0.137)	Qwen_Qwen2.5-14B-Instruct (0.134)

Table 15: Model Stability Statistics (Top-1 and Top-3 Counts)

Model	Top-1 Count	Top-3 Count
gemini-2.0-flash	2	9
gpt-4o	3	9
gemini-2.5-flash-lite	2	4
gpt-4o-mini	1	4
gemini-1.5-pro	1	3
gpt-3.5-turbo	1	3
glm-4-plus	1	2
glm-4.5	0	2
Qwen_Qwen2.5-14B-Instruct	0	1
Xianjun_PLLaMa-13b-instruct	1	1
Qwen_Qwen2.5-72B-Instruct	1	1

Table 16: Best Models at Same Scale

Size Group	Best	Second
∼7B	internlm2_5-7b-chat (0.556)	internlm_internlm2_5-7b-chat (0.529)
∼70B	Qwen_Qwen2.5-72B-Instruct (0.532)	meta-llama_Meta-Llama-3.1-70B-Instruct (0.512)

G.6 Domain-Specific Adaptation vs. General Models

To fairly answer RQ2 (whether domain models are stronger), we compare domain-specific models against open-source models of similar scale. Table 17 reports task-wise score differences (Δ = domain score − open-source average).

Findings:

- **Terminology tasks**: Domain models (especially Aksara-v1) show relative advantages on terminology-dense tasks (DS-3: Pest NER, DS-2: crop–pest matching).
- **Schema-sensitive failures**: Domain models consistently underperform on exact-span extraction (QA-3, RC-3) and structured generation (SUM-2), suggesting that domain adaptation does not automatically confer workflow competence.
- **Conclusion**: Domain adaptation $\neq$ workflow reliability. While domain knowledge helps on identification tasks, controllable output generation remains a challenge.

Table 17: Domain-Specific Models vs Open-source (Same Scale)

Domain Model	Task	Domain Score	Open-source Avg	Delta
cropinailab_aksara_v1	DS-1	0.995	0.995	+0.000
cropinailab_aksara_v1	DS-2	0.890	0.905	-0.015
cropinailab_aksara_v1	DS-3	0.000	0.341	-0.341
cropinailab_aksara_v1	QA-1	0.725	0.705	+0.020
cropinailab_aksara_v1	QA-2	0.779	0.764	+0.015
cropinailab_aksara_v1	QA-3	0.115	0.221	-0.106
cropinailab_aksara_v1	QA-4	0.196	0.191	+0.004
cropinailab_aksara_v1	RC-1	0.870	0.955	-0.085
cropinailab_aksara_v1	RC-2	0.906	0.920	-0.014
cropinailab_aksara_v1	RC-3	0.217	0.645	-0.428
cropinailab_aksara_v1	RC-4	0.434	0.390	+0.043
cropinailab_aksara_v1	SUM-1	0.196	0.121	+0.075
cropinailab_aksara_v1	SUM-2	0.100	0.079	+0.021
persadian_cropseek-llm	DS-1	0.680	0.995	-0.315
persadian_cropseek-llm	DS-2	0.375	0.905	-0.530
persadian_cropseek-llm	DS-3	0.080	0.341	-0.261
persadian_cropseek-llm	QA-1	0.175	0.705	-0.530
persadian_cropseek-llm	QA-2	0.776	0.764	+0.012
persadian_cropseek-llm	QA-3	0.003	0.221	-0.218
persadian_cropseek-llm	QA-4	0.119	0.191	-0.073
persadian_cropseek-llm	RC-1	0.325	0.955	-0.630
persadian_cropseek-llm	RC-2	0.651	0.920	-0.269
persadian_cropseek-llm	RC-3	0.029	0.645	-0.616
persadian_cropseek-llm	RC-4	0.219	0.390	-0.171
persadian_cropseek-llm	SUM-1	0.134	0.121	+0.013
persadian_cropseek-llm	SUM-2	0.009	0.079	-0.070
Xianjun_PLLaMa-13b-instruct	DS-1	0.905	0.995	-0.090
Xianjun_PLLaMa-13b-instruct	DS-2	0.865	0.905	-0.040
Xianjun_PLLaMa-13b-instruct	DS-3	0.857	0.341	+0.516
Xianjun_PLLaMa-13b-instruct	QA-1	0.600	0.705	-0.105
Xianjun_PLLaMa-13b-instruct	QA-2	0.689	0.764	-0.074
Xianjun_PLLaMa-13b-instruct	QA-3	0.000	0.221	-0.221
Xianjun_PLLaMa-13b-instruct	QA-4	0.044	0.191	-0.147
Xianjun_PLLaMa-13b-instruct	RC-1	0.815	0.955	-0.140
Xianjun_PLLaMa-13b-instruct	RC-2	0.714	0.920	-0.206
Xianjun_PLLaMa-13b-instruct	RC-3	0.008	0.645	-0.637
Xianjun_PLLaMa-13b-instruct	RC-4	0.024	0.390	-0.366
Xianjun_PLLaMa-13b-instruct	SUM-1	0.022	0.121	-0.099
Xianjun_PLLaMa-13b-instruct	SUM-2	0.000	0.079	-0.079
Xianjun_PLLaMa-7b-instruct	DS-1	0.465	0.995	-0.530
Xianjun_PLLaMa-7b-instruct	DS-2	0.355	0.905	-0.550
Xianjun_PLLaMa-7b-instruct	DS-3	0.000	0.341	-0.341
Xianjun_PLLaMa-7b-instruct	QA-1	0.000	0.705	-0.705
Xianjun_PLLaMa-7b-instruct	QA-2	0.666	0.764	-0.098
Xianjun_PLLaMa-7b-instruct	QA-3	0.005	0.221	-0.216
Xianjun_PLLaMa-7b-instruct	QA-4	0.008	0.191	-0.184
Xianjun_PLLaMa-7b-instruct	RC-1	0.395	0.955	-0.560
Xianjun_PLLaMa-7b-instruct	RC-2	0.593	0.920	-0.326
Xianjun_PLLaMa-7b-instruct	RC-3	0.011	0.645	-0.634
Xianjun_PLLaMa-7b-instruct	RC-4	0.044	0.390	-0.346
Xianjun_PLLaMa-7b-instruct	SUM-1	0.026	0.121	-0.095
Xianjun_PLLaMa-7b-instruct	SUM-2	0.000	0.079	-0.079

G.7 Key Takeaways

This detailed analysis directly addresses the three research questions:

- **RQ1 (General reasoning vs. complex IPM tasks)**: Complex IPM tasks—especially exact-span extraction, strict schema adherence, and evidence-grounded synthesis—create significant performance gaps. Bottlenecks concentrate in RC-4, SUM-2, and QA-3/RC-3 (format-sensitive cloze).
- **RQ2 (Domain models vs. general models)**: Domain-specific models are not necessarily stronger. While they may benefit on terminology/identification tasks, workflow reliability is primarily determined by controllability and instruction-following capability, where proprietary and well-aligned open-source models excel.
- **RQ3 (Scale vs. domain capability tradeoff)**: Scale generally benefits performance, but substantial variance exists at the same scale. Open-source large models (70B+) still lag significantly behind frontier proprietary models, indicating systematic headroom for improvement in instruction alignment and format control.

These findings reinforce the conclusions in Sections 4.2 and 4.3, providing quantitative evidence for workflow-level gaps, format sensitivity, and the importance of controllability in IPM applications.