

CLIMINVESTBENCH: Evaluating Agentic Investment Report Generation and Maintenance

Edward Richards
edward.richards@glasgow.ac.uk
School of Computing Science,
University of Glasgow
Glasgow, United Kingdom

Javier Sanz-Cruzado
javier.sanz-cruzado@glasgow.ac.uk
School of Computing Science,
University of Glasgow
Glasgow, United Kingdom

Richard McCreadie
richard.mccreadie@glasgow.ac.uk
School of Computing Science,
University of Glasgow
Glasgow, United Kingdom

Abstract

Generated analytical reports can inform financial decisions, so evaluation must check numerical correctness and evidential support as well as content coverage. Existing report benchmarks provide reference reports or grading criteria, but offer limited support for automatically revising expected calculations and written content when evidence is withheld or updated. Testing generation, missing inputs, and maintenance under the same reporting requirements therefore requires additional reference construction. To address this gap, we define a novel framework for evaluating the quality of structured report generation based on reusable reporting specifications that link structured evidence to required calculations and report content. Pairing a specification with different evidence conditions automatically generates both task instructions and evaluation targets, including unavailable results, revised values, and content to preserve. An automatic test suite combines deterministic checks of structure, tables, numerical results, and citations with LLM judgments of prose coverage and citation support.

Using the framework, we build a benchmark, CLIMINVESTBENCH, for evaluating the generation of bankability reports for climate mitigation projects. Our benchmark comprises 220 tasks for 20 climate-finance projects across full-evidence generation, missing-evidence generation, and report maintenance. Under this benchmark, we compare six LLMs using whole-report and section-based generation. The experiments show that high structural compliance and content coverage can coexist with numerical errors and gaps in evidential support. Section-based generation uses more recorded response time overall, but its quality benefits depend on the model and task. Specification reuse limits repeated manual preparation, while the set of separate measures expose reporting failures that a single overall score can obscure. These unknown failure modes are of particular concern where the output, as in financial report creation, contributes to high-stakes decision making.

Keywords

report generation, climate finance, LLM agents, benchmarking, missing evidence, report maintenance

1 Introduction

Structured analytical reports present evidence, calculations, and written interpretation in a prescribed format to support decision-making. In finance, examples include investment due-diligence reports [7], documented creditworthiness assessments [10], and project feasibility reports incorporating bankability analysis [1]. Readers require accurate information and a clear account of the

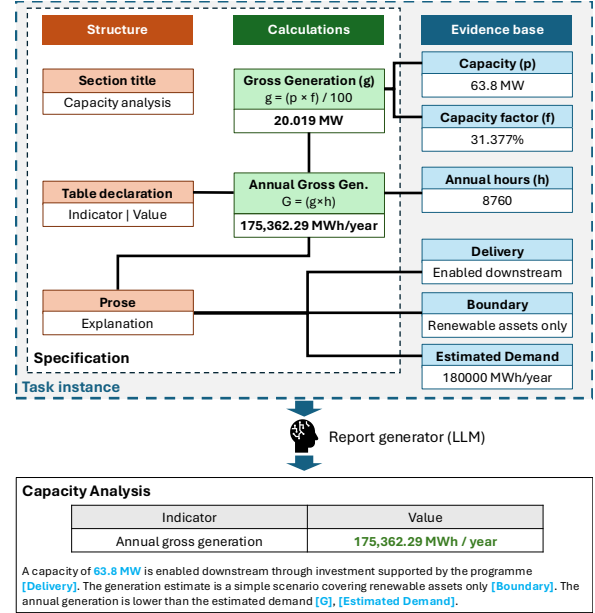


Figure 1: Example of report generation.

evidence and reasoning underlying their conclusions, because these reports can inform lending and investment decisions [2, 3]. Therefore, producing these reports require evidence, calculations, and written interpretation that remains consistent across sections, e.g. an input may appear in a table, contribute to a calculated indicator, and inform discussion across several sections. Missing inputs can make some analyses unsupported; changes to inputs can require revisions to both calculations and their interpretation. This makes constructing and maintaining such reports challenging and prone to error. Agentic AI systems have been proposed as a means to make financial analysis and report generation easier [12], and companies like DBS report deploying such systems [8]. However, while agentic AI is a promising innovation, it also introduces new failure modes, such as model hallucinations, number usage errors and faulty reasoning. As a result there is an growing need for tools and strategies to that can accurately identify errors in (increasingly generated) report content.

Based on prior works, what needs to be looked for is well understood: structural compliance [15, 28, 35, 41, 42]; numerical correctness [17, 30, 38, 42]; coverage of requested content [18, 26, 32, 38]; and evidential support [13, 18, 20, 23, 32]. However producing such

assessments is time-consuming and difficult. Human evaluation is expensive due to the significant time needed to check the report content against both reporting requirements and included supporting evidence. LLM judges can to some extent automate coverage checks and evidential support, but their judgments are subject to biases and reasoning errors [40]. Meanwhile, deterministic checks can validate the document structure, numerical values, and references, but only if provided with detailed rule-sets and/or an accurate ground-truth to compare against. None of these approaches are sufficient in isolation, but naively combining them would be prohibitively expensive, hence an approach is needed that we can be confident will detect reporting errors and inconsistencies, but is cost-effective to instantiate.

To combine these evaluation approaches while limiting repeated manual work, we present a framework for constructing reporting tasks and their evaluation targets together (Figure 2). Each task is built from a reporting specification and a structured evidence, which together form a *task instance*. The executable specification states the required sections, tables, calculations, written analysis, and citations. Reusing the same specification with different evidence creates tests for full-evidence generation, generation with missing inputs, and report maintenance after an evidence change. For each pairing, the framework automatically generates reporting instructions and evaluation targets, reducing repeated manual preparation.

An automatic test suite assesses the system’s report against the specification and the evidence supplied for its task. It combines deterministic checks of report structure, table contents, numerical results, and required citations with LLM judgments of whether the written analysis meets its requirements and whether cited evidence supports its claims.

We apply the framework to create a new climate-finance reporting dataset, CLIMINVESTBENCH¹. To do this we use 20 projects from ClimateFund [29] as evidence, and an expert-developed reporting template as the basis of our specification. Climate-project appraisal requires producing a report containing assessments of expected climate impacts, financing arrangements, and efficiency, providing a setting in which numerical indicators and written assessments must be considered together [14]. We first provide evaluation of the framework, via human audit of the LLM-as-Judge components of the evaluation. Finding high levels of agreement on judging metrics for both requirements coverage and citation support, with $\kappa = 0.725$ and (92.0%; $\kappa = 0.730$) respectively. We then compare six models under whole-report and section-based generation to examine the reporting behaviours and resource trade-offs that the benchmark exposes.

Our contributions are fourfold: (i) a framework that derives reporting tasks and evaluation targets from a common executable specification; (ii) a climate-finance dataset; (iii) an evaluation of the LLM-as-Judging component of evaluation within this climate-finance setting; and (iv) an evaluation of six LLMs and two generation strategies on climate-finance reporting tasks, showing that high structural compliance and content coverage can coexist with numerical errors and gaps in evidential support. These findings support assessing these aspects of report quality separately.

¹Benchmark code and data: <https://github.com/esric33/climinvest-bench>.

2 Related Work

Our framework brings together established approaches to report-generation benchmarking and evaluation. We review four areas that inform its design: benchmark construction, handling missing or updated evidence, deterministic and model-based evaluation, content coverage and evidential support. In each area, we explain the practices we draw on and how we integrate them through a common executable reporting specification.

Report-generation benchmarks: Report-generation benchmarks can be constructed by working backwards from existing reports or by defining reporting tasks over source evidence. The first approach reconstructs tasks and derives reference material from expert-authored outputs, as in ReportBench and AnalystBench [20, 28]. The second approach starts with source evidence and establishes what reports should contain. For example, T2R-BENCH uses model-assisted generation and human annotation to construct reporting questions and reference key points from industrial tables [38]. A subset of this second approach uses a human-authored template to instantiate instances [31]. We extend this template driven approach, formalizing the template as an executable specification that derives both task instances and the corresponding evaluation targets. Resolving its dependencies determines how expected report content changes when inputs are unavailable or updated, thereby supporting a range of evidence conditions from a single template.

Missing and updated evidence: In addition to generating reports from complete evidence, systems must handle cases where required information is missing. They should recognise when the available evidence is insufficient, a capability evaluated in retrieval-augmented question answering benchmarks such as RGB [5]. For report generation, this means making clear which parts of the report cannot be completed using the supplied information. A completed report may also need updating when its source information changes. Revision should incorporate relevant new information [21] while preserving content that remains correct. Across successive revisions, the report should also retain previously achieved content coverage and citation quality [4]. Our framework uses a common reporting specification to define what reports should contain under complete, missing, and updated evidence.

Benchmark construction also requires a way to assess completed reports. Human review can provide detailed assessment, but reviewing every output is costly across repeated experiments [39]. We distinguish two complementary approaches to automatic evaluation: deterministic checks of explicit requirements and model-based judgments of written content.

Deterministic evaluation: Explicit requirements allow reproducible checks of output structure, such as required sections and formats [41]. For numerical content, annotated calculation programs make the expected results explicit [6]. Our framework applies these principles at the report level. The reporting specification defines required document components and calculations. Applying it to the supplied evidence provides targets for checks of report structure, tables, numerical results, and required citation links.

Model-based evaluation: Written analysis requires judgments about whether requested content is included and whether cited

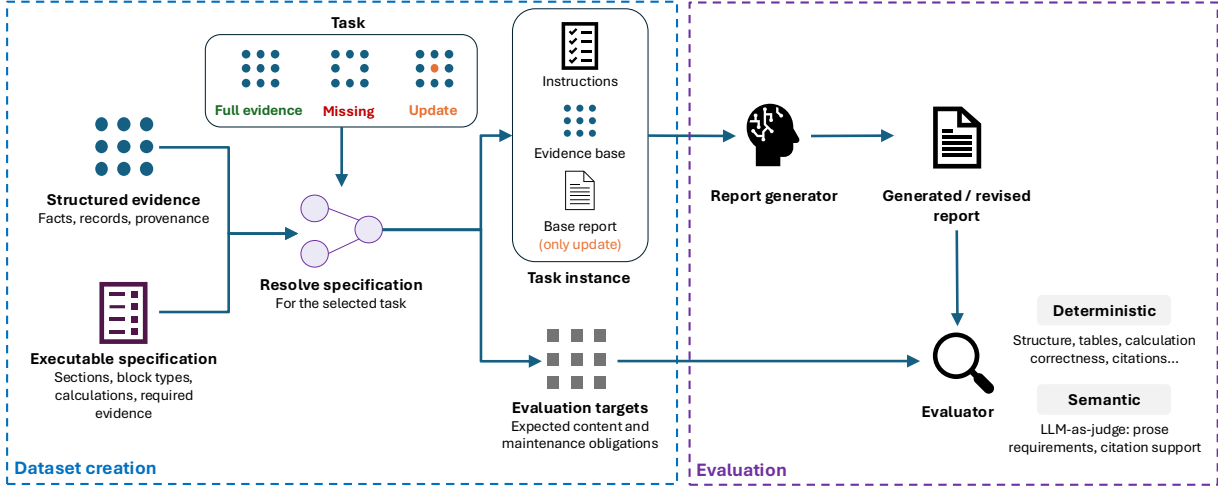


Figure 2: Overview of the proposed framework. A reporting specification and structured evidence jointly define reporting tasks and corresponding evaluation targets under full-evidence, missing-evidence, and update conditions. Generated or revised reports are assessed against these targets and supporting evidence using complementary deterministic checks and bounded semantic judgments, yielding distinct measures of report quality.

evidence supports its claims. LLM judges can assess these properties, but their judgments remain subject to bias and reasoning errors [39]. Breaking broad criteria into specific questions or checklist items makes the assessment more explicit [9, 19].

Content coverage and citation support should be assessed separately: including requested information does not establish that its citations support it. This separation is used in evidence-grounded generation and report evaluation [13, 18, 32]. We adopt these practices, assessing whether prose meets its content requirements and whether claims are supported by their cited evidence. Both assessments use requirements and evidence references derived from the reporting specification for each task.

These approaches can be combined to assess both report structure and content. FinDeepResearch evaluates report generation using references produced by models and checked by experts, combining structural checks with LLM grading of content and calculations [42]. Our executable specification derives expected values and content requirements from the supplied evidence, enabling deterministic checks of required calculation results. We assess citation support separately from content coverage. The same specification supports generation with missing evidence and report updates, including checks that unaffected content is preserved.

3 Framework Description: Dataset Creation & Evaluation

In this section, we detail the report generation tasks and our proposed framework for the creation of evaluation datasets for these. We first define reporting tasks in which a system must produce or revise a report using project evidence.

We consider three tasks: (i) generating a report from complete evidence, (ii) generating with selected inputs withheld, and (iii) updating an existing report after the evidence changes.

- **Full-evidence generation** tests whether the system produces a report that meets the specified reporting requirements.
- **Generation with missing evidence** tests whether the system completes supported analyses, identifies unavailable inputs and calculations, and explains the resulting limitations.
- **Generation with updated evidence (also known as report maintenance)** tests whether the system incorporates an evidence change and its consequences while preserving unaffected content.

In each, the system receives task instructions, descriptions of the evidence and calculations, and tools for retrieving evidence and requesting calculation results. For the update task, the system also receives the prior report and a description of the desired change. The output of all three tasks is a report containing the required sections, tables, numerical results, written analysis, and citations.

The framework derives the task instructions and evaluation targets (structural compliance, numerical correctness, coverage of requested content, evidence, etc) support from an executable specification, which both defines both what the report generator should produce and what we should look for when evaluating. An overview of the framework is shown in Figure 2.

3.1 Framework inputs and reporting specifications

Benchmark construction begins with prepared project evidence and a human-authored executable specification. The evidence contains project facts and records, together with their sources and units where applicable. The executable specification defines the required report structure, numerical results, and written analysis. Figure 3 shows a simplified executable specification for the same power

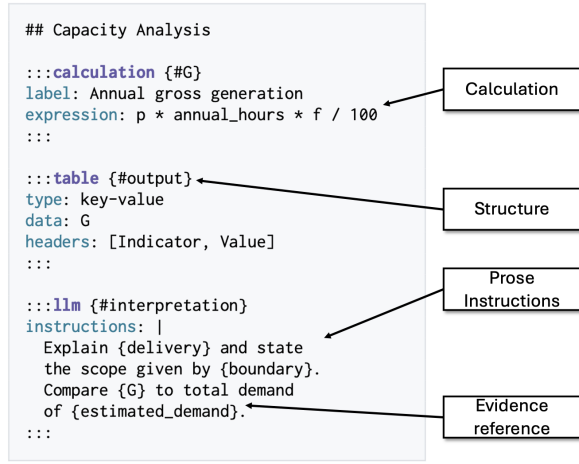


Figure 3: Simplified executable specification for a capacity-analysis section. Capacity p (MW) and capacity factor f (%) determine annual gross generation G (MWh/year). The table references this result, while the prose instruction links it to evidence about project delivery and scope.

capacity analysis example used in Figures 1 & 4. As we can see, an executable specification defines the sections of the report and information needed to either construct a task instance or evaluation targets for those sections (what data is needed, what the section should contain, such as tables or text, and any intermediate calculations).

Constructing task instances and evaluation targets: For each task instance, we combine an executable specification with project evidence for each of our three tasks. The task determines if all evidence is available and if any evidence has changed. The framework follows the specification’s links between evidence (e.g. {boundary}), calculations (e.g. G), and report content to compile into an intermediate resolved specification, which represents the report as a graph of nodes, where nodes represent sections, calculations, text blocks, and so on. From this a task instance can be deterministically produced, which is simply an LLM prompt, containing the target report structure, provided evidence and possibly a prior report. Similarly, the evaluation targets for this instance can also be derived, such as expected structure, table entries and units, calculation results or unavailable-result markers, and content and citation requirements for written analysis.

Figure 4 illustrates how the task affects the desired output report for our example. As we can see, with full evidence, capacity and capacity factor determine the annual generation value expected in the report. Withholding capacity makes the dependent calculation unavailable: the report must mark the result as unavailable and explain why it cannot be estimated. For the report updating task, increasing capacity by 5%, with the other inputs fixed, increases the expected generation value by 5% and requires the affected table entries and written analysis to be updated but unaffected sections should not be altered.

3.2 Evaluation

As discussed in Section 2, there are many ways to evaluate a report, the above framework is designed to maximise the number of diagnostic tests we can run over a report without the need for a human to manually annotate it, which we have referred to so far as our evaluation targets. Our framework supports deterministic checks to assess report structure, table content, required calculation results, declared citations and content requirements. We then add LLM-based assessment to check whether the written analysis covers its requirements and whether cited evidence supports the associated claims. In total there are 20 evaluation targets supported, however for ease of reporting we aggregate these together into 9 core metrics summarized in Table 1. The full aggregation rules, detailed formulas, matching rules, and treatment of absent or inapplicable requirements is provided in Appendix A. For an equation later we use a single-letter short-form, denoted in brackets after the metric name. The common scores S, T, K, C, R, J apply to all three tasks. Missing-evidence generation additionally uses M , while report maintenance additionally uses A and P . Scores range from zero to one, higher is better.

4 Climate-finance Benchmark

We now apply the framework to climate-finance reporting, creating CLIMINVESTBENCH, a benchmark of report generation and maintenance. It combines project evidence from ClimateFund [29] with reporting specifications adapted from an expert-developed bankability template. In this section, we describe how we prepare the evidence, adapt the template, and construct the three reporting tasks.

4.1 Project evidence and preparation

Our starting point is ClimateFund, a dataset of structured annotations from climate mitigation project proposals funded by the Green Climate Fund [29]. We use 20 of its 21 projects.² This dataset contains rich manually annotated information about the project objectives, financing, implementation, expected climate impacts, performance indicators, and risks, which we use to build complex reports for our benchmark.

We pre-process the data by standardising field names, number and unit formatting, and removing duplicate performance-indicator entries. We give individual facts and lists consistent identifiers so that reporting specifications can refer to them. We also manually extract selected values already stated in the project descriptions, recording the supporting text for each extraction.

Some reporting analyses require inputs not provided by the source records. For these, we add synthetic assumptions generated using fixed rules. These additions are explicitly labelled and kept distinct from source-supported facts. For example, a source-supported capacity value can be combined with assumptions about capacity factor and grid losses to calculate gross and net electricity generation. This lets us test how reports handle more complex dependency chains under missing or updated inputs.

²We exclude FP105 because its headline project cost is reported in EUR, while the other projects use USD. This keeps project costs in a common currency without requiring exchange-rate assumptions.

	Full evidence		Missing evidence		Update evidence	
Evidence	Capacity (p)	63.8 MW	Capacity (p)	UNKNOWN	Capacity (p)	66.99 (+5%)
	Capacity factor (f)	31.377%	Capacity factor (f)	31.377%	Capacity factor (f)	31.377%
	Delivery	Enabled downstream	Delivery	Enabled downstream	Delivery	Enabled downstream
	Estimated Demand	180,000 MWh/year	Estimated Demand	180,000 MWh/year	Estimated Demand	180,000 MWh/year
Calc.	Gross Generation (g)	20.019 MW	Gross Generation (g)	UNAVAILABLE	Gross Generation (g)	21.019 MW
	Annual Gross Gen.	175,362.29 MWh/year	Annual Gross Gen.	UNAVAILABLE	Annual Gross Gen.	184,130.40 MWh/year
Report	Capacity Analysis		Capacity Analysis		Capacity Analysis	
	Indicator	Value	Indicator	Value	Indicator	Value
	Annual gross generation	175,362.29 MWh / year	Annual gross generation	[UNAVAILABLE RESULT]	Annual gross generation	184,130.40 MWh/year
	A capacity of 63.8 MW is enabled downstream through investment supported by the programme [Delivery]. Annual Gross Generation is calculated as less than the Estimated Demand [Estimated Demand, G].		A capacity of [MISSING CAPACITY] is enabled downstream through investment supported by the programme [Delivery]. Annual Gross Generation cannot be calculated no comparison can be made [Estimated Demand] [UNAVAILABLE RESULT].		A capacity of 66.99 MW is enabled downstream through investment supported by the programme [Delivery]. Annual Gross Generation is calculated be more than the Estimated Demand [Estimated Demand, G].	

Figure 4: Applying the same reporting specification under three evidence conditions. Full evidence supports the generation estimate; withholding capacity makes the dependent result unavailable; increasing capacity changes the expected values while unaffected content is preserved.

Table 1: Evaluation categories and the finer checks or judgments they summarise.

Category	What it tests	Finer checks or judgments
Structure (S)	Whether the report follows the structure the specification declares.	Titles; section order; order of tables and prose components.
Tables (T)	Whether tables in the report have the structure and content declared by the specification.	Headers; row identities and order; cell values and units; table dimensions.
Calculated results (K)	Whether specified calculation results are reported in tables and prose.	Values and units of required results in tables and prose, including unavailable-result markers.
Citations (C)	Whether citations contained in the specification appear in the corresponding element of the report.	Table citations; required citation groups; stated values and units; required records and their order; allowed citation targets in prose.
Prose requirements coverage (R)	Whether each requirement given in the instructions of the specification for a prose block is met by the corresponding prose within the report.	A coverage judgment for each authored prose requirement, assessed in its section context.
Supported citation coverage (J)	Whether required citations support their associated claims.	Per-citation sentence level support judgments.
Missingness (M)	Whether missing inputs and unavailable results are identified.	Required unavailable-input and unavailable-result markers.
Section changes (A)	Whether the system changes sections affected by the evidence update.	For each affected section, a check that its content differs from the original report.
Section preservation (P)	Whether the system keeps sections unaffected by the evidence update unchanged.	For each unaffected section, a check that its content matches the original report.

4.2 Reporting templates and variants

We adapt an expert-developed bankability reporting template from the Horizon Europe CLIMINVEST project [11]. The original template is designed for EU-based projects (not covered by the Climate-Fund dataset) and requires more information than we have available

(like European regulatory compliance assessments or investment recommendations). Therefore, we simplify this template, keeping those sections for which we have information: project purpose, implementation, financing, impacts, and risks. In addition, different types of climate mitigation projects require different calculations

Table 2: Evidence-intervention categories used for both missing-evidence and update conditions. Collections have a single category as the only collections (risks, KPIs), contain calculations and are utilized across sections.

Evidence	Dependent calculations	Section reach
Individual value	None	One
Individual value	None	Multiple
Individual value	Present	One
Individual value	Present	Multiple
Collection	Present	Multiple

and explanations. Therefore, we also add specialist sections suited to the different project types in ClimateFund. As these sections differ for different project types, we build a different executable specification variant for each project type. Each variant has 15 sections: 11 shared sections and four specialist sections. The shared sections cover matters such as project identification, implementation arrangements, financing, monitoring, and risks, while the specialist sections address:

- **Physical assets (four projects):** capacity and electricity generation.
- **Financing facilities (13 projects):** how funds are allocated and how much additional finance they support.
- **Land-use and nature projects (three projects):** land area and carbon accounting.

Each section contains at least one table and a prose component that must explain its evidence and calculation results. For each project, its particular executable specification variant is used across all three tasks; full-evidence, missing-evidence, and maintenance.

4.3 Task construction and evidence conditions

Once we have defined the executable specifications for the different project reports, we build the task inputs and evaluation targets for our benchmark, following the procedure described in Section 3.1. For full-evidence generation, we supply each project’s prepared evidence without withholding or changing any inputs. To construct missing-evidence and maintenance tasks, we first categorise evidence in the project according to (a) whether they are an individual value or a list of them, (b) whether they are used by calculations and (c) whether they are used in one or multiple sections (either referenced directly or through calculations). The combination of these categories enables varying levels of difficulty for report generation under the missing evidence or report maintenance tasks. Table 2 sorts these categories from easier to harder to address.

For each project, we select one available and relevant input in each category. We create a missing-evidence task by withholding the selected value or entire list. We create its paired update task by changing the individual value or, if the value is a list, we remove one record, (keeping a shorter non-empty list as the updated evidence). We use fixed rules for updates and check that withholding or changing each selected input has the intended effects on calculations and report sections. Note for simplicity we use only one input per value type for this benchmark, but this is only a small portion of the missing and update task space the underlying

Table 3: Agreement with human annotations.

Assessment unit	Agreement	Cohen’s κ
Content requirement	111/118 (94.1%)	0.725
Citation occurrence	46/50 (92.0%)	0.730

evidence could support. This setup results in five missing-evidence and five maintenance tasks per project. Including the full-evidence task gives 11 task instances per project, or $20 \times (1 + 5 + 5) = 220$ task instances in total. For each task instance, the framework applies the project’s reporting specification to the available evidence to produce its instructions and evaluation targets.

For the maintenance task, we build one baseline report per project from its reporting specification and unmodified evidence set. Each maintenance task starts independently from the same baseline report, giving all systems the same starting content. We use fixed rules to fill in its headings, tables, calculation results, and citations. Its prose combines fixed statements, evidence excerpts, and summaries.

ClimateFund and the expert template provide useful starting points, but both require adaptation for the benchmark. Preparing consistent evidence records takes time, while defining suitable sections and written explanations requires domain expertise. Once this preparation is complete however, the framework uses the same specification to construct task instances and evaluation targets for each evidence set automatically. This is valuable as new test instances can be easily created by replacing part or all of the underlying evidence set.

5 Semantic evaluator verification

As described in Section 3.2, our framework uses both deterministic and LLM-based measures that draw on existing evaluation approaches and use targets derived from our reporting specification. The deterministic measures apply fixed rules, such as matching headings, comparing numerical values, and checking citation targets. However, the LLM-based measures depend on a language model to interpret whether the generated prose meets requirements and whether cited evidence supports its claims. This will be subject to some degree of error, based on how effective the LLM is at the judging task, and we therefore provide a human validation of our LLM-as-judge measures.

We use Qwen3.5-9B [34] for both LLM-based measures, with thinking disabled, temperature zero, and seed seven. The model and tokenizer revisions are fixed, with a 32,768-token context window and a 2,048-token output limit. We apply the same evaluator configuration and assessment procedure to both generation strategies. The authors assessed 118 yes/no content requirements across 30 report sections and 50 citation occurrences from the experiments described in Section 6. The sample covered all three template variants and task types, all six report-generation models, and both generation strategies. Each requirement was checked for section prose and task-specific reference facts, and each citation using its associated claim and linked evidence. Generation-model identities and automated verdicts were hidden during the annotation to avoid bias.

Table 3 reports agreement between LLM-based judging and human labels. The evaluator agreed with 111 of 118 content requirement judgments (94.1%; Cohen’s $\kappa = 0.725$) and 46 of 50 citation occurrence judgments (92.0%; $\kappa = 0.730$). We consider this to be high-enough agreement for these metrics to be useful, although we should keep in-mind that they are not perfect.

6 Experimental Design

The preceding sections define a framework for constructing reporting tasks and evaluation targets, and apply it to climate-finance reporting as CLIMINVESTBENCH. We now use this dataset to demonstrate the framework’s utility in characterising the capabilities and limitations of agentic reporting systems. Specifically, we examine how model choice and generation strategy affect report quality across full-evidence, missing-evidence, and maintenance tasks, together with the resources required to produce those reports.

6.1 Research questions

We investigate two research questions using the measures defined in Section 3.2.

- RQ1.** How do model choice and generation strategy affect: (a) initial report quality; (b) the handling of missing evidence and its downstream consequences; and (c) the correctness of updated reports and preservation of unaffected content?
- RQ2.** What trade-offs between report quality and resource use arise across these configurations?

6.2 Models

We select six models with native tool-use support and large context windows, prioritising practical deployment cost: GPT-5.6 Luna [27], Muse Spark 1.3 [22], GLM-5.3-Flash [37], MiMo-V2.5 [33], MiniMax-M3 [24], and Mistral Small 4 [25]. Each model uses the same configuration across both generation strategies. Appendix B records the model identifiers, provider routes, reasoning settings, tool-choice settings, and configured token limits.

6.3 Shared agent setup

Each model receives the reporting instructions produced during task construction (Section 3) and the complete project catalogue of evidence. The instructions cover either the complete report or an individual section, depending on the generation strategy. The catalogue includes identifiers and descriptions for all entries, including those unrelated to the immediate task, leaving the model to select the evidence and calculations it requires. For maintenance, the prompt additionally supplies the authoritative change and the relevant content from the baseline report described in Section 4.

Both strategies use an iterative agent following the ReAct interaction pattern [36]. The model retrieves evidence through `get_data` and executes declared calculations through `calculate`. These tools return values or availability information under the current evidence condition. Their observations are added to the model’s context and inform subsequent actions. A terminal tool submits the model-authored Markdown.

Each model is allowed up to eight turns (which we define together as a trajectory). The first seven permit evidence retrieval,

calculation, or submission; if the model has not already submitted the Markdown report, the eighth exposes only submission tools.

6.4 Generation strategies

Under *whole-report generation*, one trajectory constructs all 15 sections and submits the complete document through `create_report`. For maintenance, it receives the complete canonical baseline report and submits a full replacement.

Under *section-based generation*, 15 independent trajectories each construct one section and submit it through `create_section`. Their outputs are concatenated in the prescribed order. Each trajectory has its own context and calculation cache; no trajectory receives text generated by another. For maintenance, all 15 sections are considered independently. Each trajectory receives its existing section and either retains it through `keep_section` or submits revised content through `replace_section`.

The eight-turn allowance applies per trajectory, giving section-based generation a larger potential interaction budget per report. We therefore compare the quality and resource use of the complete workflows, aggregating resource use across all 15 trajectories for the section-based strategy.

The study crosses 20 projects, 11 evidence conditions, and both generation strategies. This yields $20 \times 11 \times 2 = 440$ planned report-level cases per model, or 2,640 across the six models.

6.5 Analysis protocol

Aggregation and completion. For each model, strategy, and task, we average the planned conditions within each project and then give all 20 projects equal weight. Each task mean therefore includes 20 baseline cases or 100 intervention cases per model and strategy.

Completion is reported separately from quality. A whole-report case requires an accepted report submission; a section-based case requires accepted submissions for all 15 sections.

Maintenance assessment. For maintenance, the common measures assess the submitted report against the updated evidence and reporting requirements. The affected-section change rate A and unaffected-section preservation score P additionally characterise where changes occur relative to the starting report. We interpret these measures together: changing an affected section does not establish that the revised content is correct, while preserving a section records retention of the supplied content.

Paired strategy comparisons. We compare the two strategies separately within each model and task using exact two-sided project-level sign-flip tests. Conditions are matched before their score differences are averaged within each project, giving 20 paired project observations per comparison.

Holm correction [16] is applied separately within each task, across all six models and the measures reported for that task. The families comprise 36 comparisons for full-evidence generation (S, T, K, C, R, J), 42 for missing evidence (the common measures plus M), and 48 for maintenance (the common measures plus A, P).

Composite quality. The separate measures remain primary for RQ1. For the quality–resource comparison in RQ2, we additionally

define an equal-weight mean of the six measures shared by all tasks:

$$Q_{\text{report}} = \frac{S + T + K + C + R + J}{6}. \quad (1)$$

We do this to produce a single metric which supports quality–resource comparisons to illustrate the resource–quality tradeoff. Such tradeoff analysis could be carried out with any of the individual metrics.

Resource accounting. Our primary time measure is the sum of recorded model-response durations, including responses from unsuccessful generations and all trajectories under section-based generation. It includes responses returned to the runner whether or not they lead to an accepted report. We additionally report token use, returned model-response turns, executed tool calls, and estimated generation-response costs at current standard uncached prices in Appendices D, E, F and, C respectively.

7 Results

We first examine how model choice and generation strategy affect report quality across baseline generation, generation with missing evidence, and report maintenance (RQ1). We then assess the tradeoffs between report quality and resource use (RQ2).

7.1 RQ1(a): Full-evidence generation

Table 4 compares whole-report and section-based generation when all evidence is available. Scores average over all 20 projects, including incomplete reports under the failure policy defined in Section 6.

We observe from Table 4 that a generated report can demonstrate high prose coverage but relatively poor result correctness and citation support. Under section-based generation, GPT-5.6 Luna and Muse Spark 1.3 achieved similar coverage scores (*Prose Req.*=98.4 and 98.5), while their calculated-result scores (*Calc. Results*) were 68.1 and 88.6, and their support scores (*Cite. Coverage*) were 57.7 and 72.7, respectively. Muse achieved the highest observed calculated-result scores under both strategies (88.6 for section-based and 89.2 for whole-report generation). Its section-based configuration also achieved the highest citation (*Cite. Presence*) and support (*Cite. Coverage*) scores. These comparisons show that similar coverage of the requested prose can accompany different levels of compliance with calculation and evidence-support requirements.

The benefits of section-based generation varied across models and measures. Seven of the 36 paired comparisons remained significant after Holm correction, all favouring section-based generation: table (*Table*) and citation (*Cite. Presence*) scores for Luna; citation (*Cite. Presence*) and support (*Cite. Coverage*) scores for Muse; and citation (*Cite. Presence*), requirements coverage (*Prose Req.*), and support (*Cite. Coverage*) scores for Mistral Small 4. The largest of these differences were Mistral’s prose coverage (93.0 versus 72.4) and Luna’s citation-obligation score (71.6 versus 52.9). No difference in calculated-result (*Calc. Results*) correctness between strategies remained significant after correction. The observed means do not show a uniform advantage for section-based generation, although the exceptions: GLM and MiMo, were not significant different after correction.

7.2 RQ1(b): Generation with missing evidence

Table 5 reports results across the five missing-evidence conditions, averaged within each project before aggregation. The common measures assess reports against the requirements resolved under each condition, while *Missingness* assesses identification of unavailable inputs and dependent results through the prescribed missingness markers.

Muse Spark 1.3 achieved the highest observed missingness scores under both strategies (92.2 for section-based and 94.2 for whole-report generation), alongside calculated-result scores of 90.6 and 89.8. Other configurations showed a clearer separation between prose coverage and missingness handling. MiMo V2.5 achieved *Prose Req.*=98.3 under section-based generation and 97.6 under whole-report generation, while its corresponding missingness scores were 45.7 and 60.2. High coverage of the resolved prose requirements therefore coexisted with incomplete or incorrect identification of the required unavailable-value targets.

The preferred strategy for missingness handling differed across models. Whole-report generation achieved significantly higher *Missingness* for GLM 5.3 Flash (64.0 versus 40.3) and MiMo (60.2 versus 45.7), whereas section-based generation achieved significantly higher *Missingness* for Mistral Small 4 (42.6 versus 23.3).

Across all seven measures, 21 of the 42 paired comparisons remained significant after correction: 16 favoured section-based generation and five favoured whole-report generation. Section-based generation achieved higher scores on all seven measures for Mistral, including prose coverage of 92.3 compared with 60.6. Its absolute calculated-result and citation-support scores nevertheless remained low (*Calc. Results*=42.0 and *Cite. Coverage*=9.3). For Luna, section-based generation produced significant advantages in tables, calculated results, citation obligations, and citation support. For Muse, its significant advantages concerned structure, tables, citation obligations, prose coverage, and citation support.

Completion also contributed to the observed workflow differences. GLM completed 89 of 100 cases under section-based generation and all 100 under whole-report generation; its whole-report advantages were significant for structure, calculated results, citation support, and missingness. MiniMax completed 97 cases under section-based generation and 86 under whole-report generation. Although its section-based means were higher on every measure, none of these differences remained significant after correction.

7.3 RQ1(c): Report maintenance

Table 6 reports maintenance performance across the five authoritative-update conditions. The common measures assess the resulting reports against the updated requirements, while *Sec. Changes* and *Sec. Preservation* distinguish changes to affected sections from preservation of unaffected sections.

Affected-section change rates were generally high: eight of the 12 model–strategy configurations achieved *Sec. Changes*=100. However, preservation varied substantially, including among configurations that changed every affected section. These outcomes distinguish whether editing occurred in the required sections from whether it extended to unrelated content. Eighteen of the 48 paired comparisons remained significant after Holm correction, with 11

Table 4: Full-evidence generation. Scores on a 0–100 scale. Bold marks the higher unrounded mean within each model.

Model	Strategy	Deterministic				Semantic	
		Structure	Table	Calc. Results	Cite Presence	Prose Req.	Cite Coverage
GPT-5.6 Luna	Section Report	99.8	94.6*	68.1	71.6*	98.4	57.7
		98.7	88.9	56.8	52.9	98.4	54.4
Muse Spark 1.3	Section Report	100.0	98.6	88.6	86.2*	98.5	72.7*
		97.5	97.9	89.2	78.1	97.8	59.1
GLM 5.3 Flash	Section Report	89.9	83.2	57.3	55.8	89.1	50.7
		100.0	90.3	65.7	59.1	99.1	62.9
MiMo V2.5	Section Report	94.6	89.4	68.7	65.6	94.2	52.7
		97.6	96.0	70.9	69.1	98.4	61.8
MiniMax M3	Section Report	89.9	85.4	66.2	65.5	89.0	54.6
		88.5	84.2	60.1	63.0	88.6	60.1
Mistral Small 4	Section Report	99.0	87.1	43.9	39.9*	93.0*	8.4*
		90.7	77.8	32.3	25.1	72.4	2.1

All 20 projects contribute equally. * $p_{\text{Holm}} < .05$ indicates significant difference between section and report level generation for that metric.

Table 5: Missing evidence. Scores on a 0–100 scale. Bold marks the higher unrounded mean within each model.

Model	Strategy	Deterministic				Semantic		Task-specific
		Structure	Table	Calc. Results	Cite Presence	Prose Req.	Cite Coverage	Missingness
GPT-5.6 Luna	Section Report	99.8	93.8*	66.7*	71.8*	98.6	59.0*	73.7
		100.0	89.4	58.9	55.2	98.1	48.7	71.1
Muse Spark 1.3	Section Report	100.0*	98.0*	90.6	86.1*	98.6*	72.3*	92.2
		93.7	97.0	89.8	78.7	97.6	61.3	94.2
GLM 5.3 Flash	Section Report	88.9	81.5	58.0	55.7	88.4	51.2	40.3
		100.0*	90.3	67.4*	58.8	98.9	63.0*	64.0*
MiMo V2.5	Section Report	99.7	92.9	71.4	68.8	98.3	53.7	45.7
		98.0	94.1	68.6	67.2	97.6	55.7	60.2*
MiniMax M3	Section Report	96.9	90.9	70.4	69.1	96.0	59.0	61.2
		84.0	80.0	57.7	60.3	83.7	57.4	51.6
Mistral Small 4	Section Report	99.2*	87.0*	42.0*	41.1*	92.3*	9.3*	42.6*
		78.2	64.2	23.5	21.0	60.6	0.8	23.3

All 20 projects contribute equally. * $p_{\text{Holm}} < .05$ indicates significant difference between section and report level generation for that metric.

favouring section-based generation and seven favouring whole-report generation. None of the strategy differences in *Sec. Changes* was significant.

Section-based generation achieved significantly higher preservation for GPT-5.6 Luna (99.6 versus 0.0) and MiMo V2.5 (100.0 versus 86.1). Luna’s whole-report workflow retained no unaffected section unchanged under the parsed-content comparison, despite changing every affected section. Mistral Small 4 showed the reverse preservation pattern: whole-report generation achieved *Sec. Preservation*=70.2, compared with 37.6 for section-based generation. Preservation therefore depended on both the model and generation strategy.

Report quality also differed independently of preservation. Luna’s whole-report workflow achieved significantly higher prose coverage than its section-based workflow (97.1 versus 66.6), while section-based generation achieved significantly higher calculated-result correctness (99.7 versus 97.3) and citation support (64.0 versus 55.8). For Mistral, section-based generation achieved higher

prose coverage (77.4 versus 62.9), whereas whole-report generation achieved higher citation support (60.9 versus 57.4), alongside its higher preservation. Each of these differences remained significant after correction.

Completion affected some of the maintenance scores. MiniMax M3 completed 89 of 100 section-based cases and 96 whole-report cases, matching its respective affected-section change rates of 89.0 and 96.0. GLM 5.3 Flash completed 99 section-based cases, corresponding to *Sec. Changes*=99.0. In each of these configurations, every accepted report changed all affected sections; incomplete reports account for the remaining deficit in *Sec. Changes*.

7.4 RQ2: Report quality and resource use

Figure 5 compares the composite quality score with summed recorded model-response time for each task.

Table 6: Report maintenance. Scores on a 0–100 scale. Bold marks the higher unrounded mean within each model.

Model	Strategy	Deterministic				Semantic		Task-specific	
		Structure	Table	Calc. Results	Cite Presence	Prose Req.	Cite Coverage	Sec. Changes	Sec. Preservation
GPT-5.6 Luna	Section	100.0	99.9*	99.7*	81.8*	66.6	64.0*	100.0	99.6*
	Report	100.0	99.5	97.3	77.8	97.1*	55.8	100.0	0.0
Muse Spark 1.3	Section	100.0	100.0	99.9	81.9	66.1*	63.8*	100.0	100.0
	Report	100.0	100.0	100.0	83.1*	58.7	63.2	100.0	99.0
GLM 5.3 Flash	Section	99.0	99.0	98.9	81.1	63.4	63.2	99.0	98.7
	Report	100.0*	100.0	99.9	82.6*	66.6	63.7	100.0	82.0
MiMo V2.5	Section	99.9	100.0*	99.8	82.4	62.9*	63.5	100.0	100.0*
	Report	98.7	99.9	99.9	83.0*	59.3	63.4	100.0	86.1
MiniMax M3	Section	89.0	89.0	88.8	72.7	57.6	56.5	89.0	89.0
	Report	96.0	96.0	95.8	79.5	57.5	60.9	96.0	91.2
Mistral Small 4	Section	98.8	98.8	95.7	77.9	77.4*	57.4	98.0	37.6
	Report	99.4	98.4	95.6	80.1	62.9	60.9*	100.0	70.2*

All 20 projects contribute equally. * $p_{\text{Holm}} < .05$ indicates significant difference between section and report level generation for that metric.

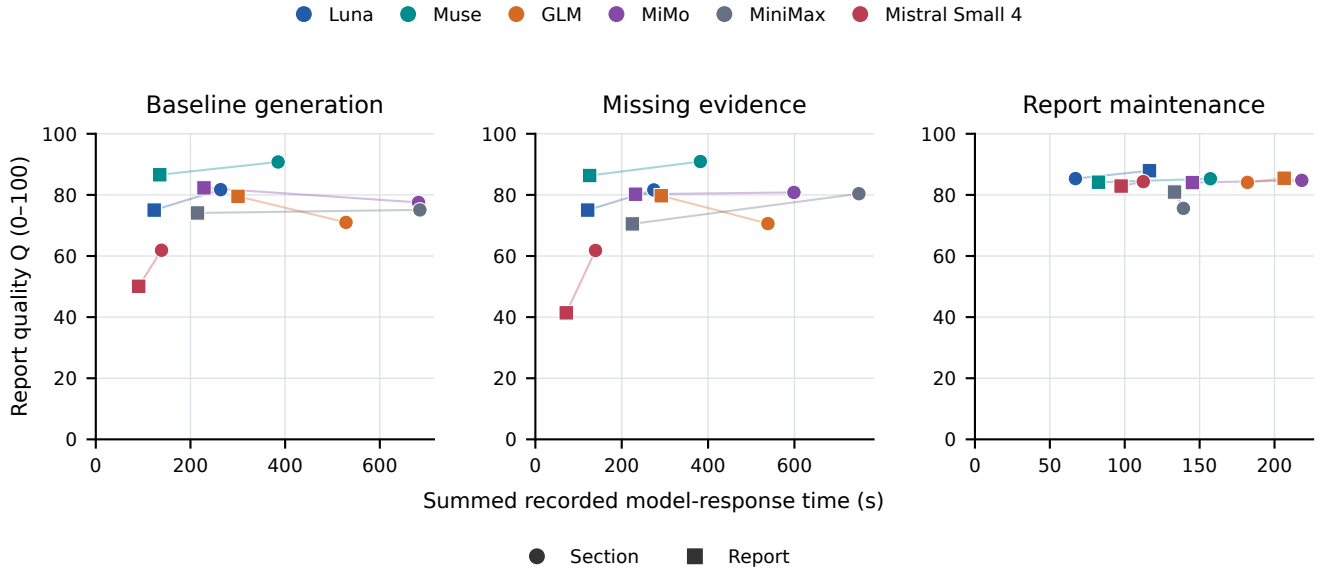


Figure 5: Report quality and summed recorded model-response time across the three tasks. Quality is the equal-weight mean $Q = (S + T + K + C + R + J)/6$.

Across the complete task mixture, section-based generation used between 1.5 and 2.6 times the recorded response time of whole-report generation. The additional resource use accompanied different quality outcomes. Muse Spark 1.3 achieved the highest observed overall quality with section-based generation (*Composite Quality*=88.3), compared with 85.4 for whole-report generation, at respective mean response times of 280.2 and 106.9 seconds. Mistral Small 4 showed the largest increase in composite quality between strategies (72.1 versus 61.1), with times of 127.0 and 85.1 seconds.

For other models, additional time brought little or no improvement in the composite. MiMo V2.5 achieved similar overall quality under section-based and whole-report generation (82.3 and 82.1), while section-based generation used 433.3 seconds compared with

192.2 seconds. GLM 5.3 Flash had both lower quality and higher recorded response time under section-based generation: *Composite Quality*=76.8 at 375.5 seconds, compared with 82.3 at 253.7 seconds. The larger potential interaction budget of the section-based workflow therefore did not consistently translate into higher report quality.

The relationship differed by task. For baseline and missing-evidence generation, section-based workflows used more recorded response time for every model. Maintenance provided exceptions: Luna’s section-based workflow used 67.1 seconds compared with 116.5 seconds for whole-report generation, while GLM used 181.9 compared with 206.5 seconds. Their section-based composite scores

Table 7: Quality and recorded response time over the complete 1:5:5 task mix.

Model	Strategy	Q (0–100)	Time (s)	Cases
GPT-5.6 Luna	Section	83.3	179.2	220
	Report	80.9	119.3	220
Muse Spark 1.3	Section	88.3	280.2	220
	Report	85.4	106.9	220
GLM 5.3 Flash	Section	76.8	375.5	220
	Report	82.3	253.7	220
MiMo V2.5	Section	82.3	433.3	220
	Report	82.1	192.2	220
MiniMax M3	Section	77.7	466.1	220
	Report	75.6	182.2	220
Mistral Small 4	Section	72.1	127.0	220
	Report	61.1	85.1	220

$Q = (S + T + K + C + R + J) / 6$. Conditions are averaged within project and the 20 projects receive equal weight. Time sums recorded model-response durations across all trajectories; it is not elapsed report-production time.

were lower, however: 85.3 versus 87.9 for Luna and 84.1 versus 85.5 for GLM.

8 Discussion

The results show why report quality cannot be judged from structure or content coverage alone. A report can follow the required format and discuss the requested topics while still reporting incorrect calculation results or citing evidence that does not support its claims. For reports used to inform financial decisions, these weaknesses matter even when the report appears complete. Separate measures make them visible and identify which reporting requirements remain unmet.

The workflow results also show substantial variance depending on what is being measured. Section-based generation improves some outcomes, but its benefits vary across models and tasks, and its additional resource use does not consistently produce higher quality. Workflow choice therefore depends on both reporting requirements and available resources. An overall quality score provides a compact summary for resource comparisons, while the individual scores reveal weaknesses that this average can hide, including missing-input handling and preservation of unchanged content.

Reusing the same reporting requirements across evidence conditions allows us to test capabilities beyond initial report generation. The results show that covering the requested topics does not guarantee that missing inputs and unavailable results are identified. Similarly, changing sections affected by an update can coincide with unnecessary changes elsewhere. The specification links inputs to calculations and report content, allowing instructions and evaluation targets to be generated for each condition without defining them separately each time. Maintenance still requires several measures: a changed section may contain errors, so change and preservation scores must be read alongside the scores for the resulting report’s content.

The framework is best suited to recurring reports that share a common format and clear links between evidence, calculations, and written analysis. Projects can differ, provided their evidence can be organised into a common structure. Preparing this evidence

takes time: records need consistent fields, units, identifiers, and source references. Adapting a specification also requires domain expertise to declare appropriate calculations and instructions for explanations. Once this work is complete however, the evidence structure and reporting template can be reused across projects and evidence conditions.

This study covers 20 climate-finance projects, three variants of one expert template, and controlled changes to prepared evidence. The synthetic assumptions support selected calculations, so success on these tasks does not establish complete investment-appraisal capability. Maintenance results also depend on the supplied baseline: some weaknesses in its prose and citation support may be retained when unaffected content is preserved. The small, single-author audit provides positive evidence about the model-based metrics within this setting, however further applications and assessments are needed to establish how well this transfers to other reporting settings.

9 Conclusion

We presented a framework that pairs reusable reporting templates with structured evidence to construct tasks and evaluation targets for full-evidence generation, missing inputs, and report maintenance. Evaluation combines deterministic checks with LLM judgments of prose coverage and citation support. Tests with six models on CLIMINVESTBENCH showed that high structural compliance and prose coverage can coexist with incorrect calculation results and weak citation support. Section-based generation improved some outcomes, but its benefits varied across models and tasks. By reusing reporting requirements and assessing quality through separate measures, the framework enables consistent comparisons of systems and reveals failures that a single overall score can hide.

Acknowledgements

This work has been supported by the CLIMINVEST project, funded by the European Union through the Horizon Europe programme under grant agreement No. 101212947.

References

- [1] Asian Development Bank, European Bank for Reconstruction and Development, Inter-American Development Bank, Islamic Development Bank, and World Bank Group. 2016. *The APMG Public-Private Partnership (PPP) Certification Guide*. World Bank Group, Washington, DC. Chapter 4: Appraising PPP Projects. <https://ppp.worldbank.org/sites/default/files/2025-04/chapter-4-appraising-ppp-projects.pdf>
- [2] Basel Committee on Banking Supervision. 2013. *Principles for Effective Risk Data Aggregation and Risk Reporting*. Guidelines. Bank for International Settlements. <https://www.bis.org/publications/201301-guidelines-principles-effective-risk-data-aggregation-and-risk-reporting.pdf>
- [3] CFA Institute. 2024. Standard V(B): Communication with Clients and Prospective Clients. Updated January 2024; accessed 8 October 2026. <https://www.cfainstitute.org/standards/professionals/code-ethics-standards/standards-of-practice-v-b>
- [4] Bingsen Chen, Boyan Li, Ping Nie, Yuyu Zhang, Xi Ye, and Chen Zhao. 2026. Beyond Single-shot Writing: Deep Research Agents are Unreliable at Multi-turn Report Revision. arXiv:2601.13217 [cs.CL] <https://arxiv.org/abs/2601.13217>
- [5] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 38. 17754–17762.
- [6] Zhiyu Chen, Wenhui Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in*

- Natural Language Processing*. Association for Computational Linguistics, 3697–3711. doi:10.18653/v1/2021.emnlp-main.300
- [7] Cliffwater LLC. 2011. *Hedge Fund Investment Due Diligence Report: P&S Credit Management, L.P.* Investment due diligence report. Cliffwater LLC. <https://data.treasury.ti.gov/dataset/96dcb86f-e97e-4b05-8ce2-a40289e477a6/resource/d045e94f-a752-4f59-8590-9b57b13aafb6/download/gracie-investment-due-diligence-report-sept-2011redacted.pdf>
 - [8] DBS Bank. 2026. DBS Scales Agentic AI to Transform Way of Working for Corporate Bankers, Freeing Up Time for More Strategic Client Engagements. News release. Published 19 August 2026; accessed 8 October 2026. https://www.dbs.com/newsroom/DBS_scales_agentic_AI_to_transform_way_of_working_for_corporate_bankers_freeing_up_time_for_more_strategic_client_engagements
 - [9] Laura Dietz. 2024. A Workbench for Autograding Retrieve/Generate Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 1963–1972. doi:10.1145/3626772.3657871
 - [10] European Banking Authority. 2020. *Guidelines on Loan Origination and Monitoring*. Guidelines EBA/GL/2020/06. European Banking Authority. <https://www.eba.europa.eu/activities/single-rulebook/regulatory-activities/credit-risk/guidelines-loan-origination-and-monitoring>
 - [11] European Commission. 2026. CLIMINVEST: Bankable by Design, Continuous, and Predictive Climate Adaptation Investments with Co-Benefits. CORDIS project fact sheet. Grant agreement No. 101212947. Accessed 9 October 2026. doi:10.3030/101212947
 - [12] Sumitra Ganesh, Leo Ardon, Daniel Borrajo, Deepeka Garg, Udari Madhushani Sewhaw, Annapoorani Lakshmi Narayanan, Giuseppe Canonaco, and Manuela M. Veloso. 2025. Generative AI Agents for Knowledge Work Augmentation in Finance. *Annual Review of Control, Robotics, and Autonomous Systems* 8 (2025), 189–210. doi:10.1146/annurev-control-022823-033813
 - [13] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling Large Language Models to Generate Text with Citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 6465–6488. doi:10.18653/v1/2023.emnlp-main.398
 - [14] Green Climate Fund. 2015. *Initial Investment Framework: Activity-Specific Sub-Criteria and Indicative Assessment Factors*. Technical Report GCF/B.09/23, Annex III. Green Climate Fund. Adopted by Decision B.09/05(b). <https://www.greenclimate.fund/document/initial-investment-framework-activity-specific-sub-criteria-and-indicative-assessment>
 - [15] Janghoon Han, Heegyu Kim, Changho Lee, Dahm Lee, Min Hyung Park, Hosung Song, Stanley Jungkyu Choi, Moontae Lee, and Honglak Lee. 2026. DEER: A Benchmark for Evaluating Deep Research Agents on Expert Report Generation. In *Forty-third International Conference on Machine Learning*. <https://openreview.net/forum?id=ILRx5neJk6>
 - [16] Sture Holm. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70. <http://www.jstor.org/stable/4615733>
 - [17] Song Jin, Shuqi Li, Shukun Zhang, and Rui Yan. 2026. FinRpt: Dataset, Evaluation System and LLM-based Multi-agent Framework for Equity Research Report Generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. AAAI Press, 507–515. doi:10.1609/aaai.v40i1.37014
 - [18] Dawn Lawrie, Sean MacAvaney, James Mayfield, Luca Soldaini, Eugene Yang, and Andrew Yates. 2026. Overview of the TREC 2025 RAGTIME Track. arXiv:2602.10024 <https://arxiv.org/abs/2602.10024>
 - [19] Yuyang Lee, JoongHoon Kim, Jaehye Kim, Hyowon Cho, Jaewook Kang, Pil-sung Kang, and Najoung Kim. 2025. CheckEval: A reliable LLM-as-a-Judge framework for evaluating text generation using checklists. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (Eds.). Association for Computational Linguistics, Suzhou, China, 15771–15798. doi:10.18653/v1/2025.emnlp-main.796
 - [20] Minghao Li, Ying Zeng, Zhihao Cheng, Cong Ma, and Kai Jia. 2025. ReportBench: Evaluating Deep Research Agents via Academic Survey Tasks. arXiv:2508.15804 <https://arxiv.org/abs/2508.15804>
 - [21] Robert L. Logan IV, Alexandre Passos, Sameer Singh, and Ming-Wei Chang. 2022. FRUIT: Faithfully Reflecting Updated Information in Text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 3670–3686. doi:10.18653/v1/2022.naacl-main.269
 - [22] Meta. 2026. Introducing Muse Spark 1.3. Accessed 9 October 2026. <https://research.meta.ai/blog/introducing-muse-spark-1-3>
 - [23] Sewon Min, Kalpesh Krishna, Xinxin Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 12076–12100. doi:10.18653/v1/2023.emnlp-main.741
 - [24] MiniMax. 2026. MiniMax-M3 Model Card. Accessed 9 October 2026. <https://huggingface.co/MiniMaxAI/MiniMax-M3>
 - [25] Mistral AI. 2026. Introducing Mistral Small 4. Accessed 9 October 2026. <https://mistral.ai/news/mistral-small-4/>
 - [26] Ani Nenkova and Rebecca Passonneau. 2004. Evaluating Content Selection in Summarization: The Pyramid Method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*. Association for Computational Linguistics, 145–152. <https://aclanthology.org/N04-1019/>
 - [27] OpenAI. 2026. GPT-5.6 Luna Model. Accessed 9 October 2026. <https://developers.openai.com/api/docs/models/gpt-5.6-luna>
 - [28] Chau Minh Pham, Zichao Wang, Puneet Mathur, Alexa Siu, Akriti Jain, Aparna Garimella, Ananya B. Sai, Nedim Lipka, Mohit Iyyer, and Varun Manjunatha. 2026. AnalystBench: Benchmarking professional long-form report generation with web-mined multimodal tasks. In *Findings of the Association for Computational Linguistics: ACL 2026*. Association for Computational Linguistics, 23894–23926. doi:10.18653/v1/2026.findings-acl.1197
 - [29] Javier Sanz-Cruzado Puig, Miruna Clinciu, Richard McCreadie, Craig Macdonald, and Iadh Ounis. n.d. ClimateFund: An Annotated Dataset of Climate Mitigation Projects for Supporting Question Answering. Hugging Face dataset. Dataset card accessed 7 September 2026. The source revision is recorded in the benchmark manifest. doi:10.57967/hf/9046
 - [30] Craig Thomson and Ehud Reiter. 2020. A Gold Standard Methodology for Evaluating Accuracy in Data-To-Text Systems. In *Proceedings of the 13th International Conference on Natural Language Generation*. Association for Computational Linguistics, 158–168. doi:10.18653/v1/2020.inlg-1.22
 - [31] Yong-En Tian, Yu-Chien Tang, Kuang-Da Wang, An-Zi Yen, and Wen-Chih Peng. 2025. Template-Based Financial Report Generation in Agentic and Decomposed Information Retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (Padua, Italy) (SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 2706–2710. doi:10.1145/3726302.3730253
 - [32] William Walden, Marc Mason, Orion Weller, Laura Dietz, John Conroy, Neil Molino, Hannah Recknor, Bryan Li, Gabrielle Kaili-May Liu, Yu Hou, Dawn Lawrie, James Mayfield, and Eugene Yang. 2025. Auto-ARGUE: LLM-Based Report Generation Evaluation. Accepted to the SIGIR 2026 Demo Track. arXiv:2509.26184 <https://arxiv.org/abs/2509.26184>
 - [33] Xiaomi MiMo. 2026. MiMo-V2.5 Model Card. Accessed 9 October 2026. <https://huggingface.co/XiaomiMiMo/MiMo-V2.5>
 - [34] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianhao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] <https://arxiv.org/abs/2505.09388>
 - [35] Jialin Yang, Dongfu Jiang, Lipeng He, Sherman Siu, Yuxuan Zhang, Disen Liao, Zhuofeng Li, Huaye Zeng, Yiming Jia, Haozhe Wang, Benjamin Schneider, Chi Ruan, Wentao Ma, Zhiheng Lyu, Yifei Wang, Yi Lu, Quy Duc Do, Ziyao Jiang, Ping Nie, and Wenhui Chen. 2026. StructEval: Benchmarking LLMs' Capabilities to Generate Structural Outputs. *Transactions on Machine Learning Research* (2026). <https://openreview.net/forum?id=buDwV7LUA7>
 - [36] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*. <https://arxiv.org/abs/2210.03629>
 - [37] Z.ai. 2026. GLM-5.3-Flash Model Card. Accessed 9 October 2026. <https://huggingface.co/zai-org/GLM-5.3-Flash>
 - [38] Jie Zhang, Changzai Pan, Sishi Xiong, Kaiwen Wei, Yu Zhao, Xiangyu Li, Jiaxin Peng, Xiaoyan Gu, Jian Yang, Wenhan Chang, Zhenhe Wu, Jiang Zhong, Shuangyong Song, and Xuelong Li. 2025. T2R-BENCH: A Benchmark for Real World Table-to-Report Task. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 22427–22451. doi:10.18653/v1/2025.emnlp-main.1141
 - [39] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems* 36 (2023), 46595–46623.
 - [40] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL] <https://arxiv.org/abs/2306.05685>

- [41] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-Following Evaluation for Large Language Models. arXiv:2311.07911 [cs.CL] <https://arxiv.org/abs/2311.07911>
- [42] Fengbin Zhu, Xiang Yao Ng, Ziyang Liu, Chang Liu, Xianwei Zeng, Chao Wang, Tianhui Tan, Xuan Yao, Pengyang Shao, Min Xu, Zixuan Wang, Jing Wang, Xin Lin, Junfeng Li, Jingxian Zhu, Yang Zhang, Wenjie Wang, Fuli Feng, Richang Hong, Huanbo Luan, Ke-Wei Huang, and Tat-Seng Chua. 2025. FinDeepResearch: Evaluating Deep Research Agents in Rigorous Financial Analysis. arXiv:2510.13936 <https://arxiv.org/abs/2510.13936>

A Metric definitions and matching rules

This appendix gives the definitions underlying the category scores in Table 1. Equal weights apply within each specified average, including nested averages; the categories do not imply a flat average over all underlying checks. The individual measures remain available for separate reporting.

A.1 Deterministic assessment

We parse the submitted Markdown into sections, headings, tables, prose components, and citation links, then align them with the required document components. Sections are identified by their headings and tables by their position within a section. Incompatible types cannot match. Prose is aligned by component type, without using its claims or citations to choose a match; standalone headings and supplied table-title captions are excluded from prose matches. Ties prefer an available optimal match, then skipping an observed block, then skipping an expected component.

Several measures use the matching function

$$F(m, n_o, n_e) = \begin{cases} 2m/(n_o + n_e), & n_o + n_e > 0, \\ 1, & n_o = n_e = 0, \end{cases} \quad (2)$$

where m is the matched count and n_o, n_e are the observed and expected counts. Sequence matching uses the length of the longest common subsequence (LCS); set matching uses intersection size. Both omissions and additional items can therefore reduce the score. Unless stated otherwise, a proportion with no applicable obligations is one, indicating an absence of violations.

Structure (S). We assess the report title, section order, component sequence, and table presence:

$$S = \frac{h + f_{\text{sec}} + f_{\text{blk}} + f_{\text{tab}}}{4}. \quad (3)$$

Here h indicates an exact title match; f_{sec} and f_{blk} are LCS F1 scores for section identifiers and section-labelled component sequences. For f_{tab} , the matched count is the sum of the smaller of the observed and required table counts in each required section; Equation 2 then compares this with the total observed and expected counts. S measures conformity to the prescribed organisation.

Tables (T). Tables are assessed for headers, row identities and order, cell content, and dimensions. Rows are matched using their first-cell citation target, or their normalised label when no citation is present. Cell comparison indexes rows by this key; if a key is repeated, the last row with that key is used. Row-order scoring uses the complete sequence of row keys.

For text comparison, Markdown links are replaced by their visible labels, whitespace is collapsed, HTML entities are decoded, and the result is case-folded. Numeric comparison parses signed decimal quantities, removing thousands-separator commas, and compares

their decimal values exactly. Parsed unit and currency strings retain their case and must match exactly. Matching does not convert units or apply a numerical tolerance.

For expected value occurrences X , let X_{num} contain the occurrences whose expected values are numeric. Define

$$v(X) = \frac{N_{\text{value match}}(X)}{|X|}, \quad u(X) = \frac{N_{\text{unit match}}(X_{\text{num}})}{|X_{\text{num}}|}. \quad (4)$$

The numerators count occurrences that pass the corresponding comparison. An occurrence may be a table cell or an explicitly required value in prose. An absent or unmatched occurrence fails each applicable comparison. A fraction with no applicable occurrences is one. The combined value-and-unit score is

$$V(X) = \begin{cases} (v(X) + u(X))/2, & \text{if numeric values occur,} \\ v(X), & \text{otherwise.} \end{cases} \quad (5)$$

Let $\mathcal{B}$ be the set of required tables. For table $b \in \mathcal{B}$, let X_b contain its expected cells and define

$$t_b = \frac{h_b + f_b + V(X_b) + d_b}{4}, \quad T = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} t_b, \quad (6)$$

where f_b is row-identity LCS F1. The header score h_b is one only when the parsed observed and expected header lists are identical. The dimension score d_b is one only when the row count is correct and every observed row has the expected number of columns. Formally, with header lists $H_o(b), H_e(b)$ and row lists $R_o(b), R_e(b)$,

$$h_b = \mathbf{1}[H_o(b) = H_e(b)], \quad (7)$$

$$d_b = \mathbf{1}[|R_o(b)| = |R_e(b)| \wedge \forall r \in R_o(b) : |r| = |H_e(b)|]. \quad (8)$$

Missing tables receive zero, as do missing expected cells in the value and unit comparisons. Required tables receive equal weight; with no required tables, $T = 1$.

In the running example, the generation row can be present and correctly labelled while its value or unit is wrong; these are distinct contributions to T .

Calculated-result correctness (K). Let X_{calc} contain the required occurrences of calculated results in tables and prose. Required occurrences are matched one-to-one by calculation identifier and report component, in order where repeated. Prose values and units are read from the labels of their calculation citations. Value and unit matches also require the expected citation target, including its availability state. We exclude row labels and values taken directly from source evidence. Available non-scalar prose outputs, such as record lists, are assessed through selection and coverage rather than this scalar-result measure. We compute

$$K = V(X_{\text{calc}}). \quad (9)$$

Value and unit counts are pooled across the required occurrences, including required unavailable-result markers. An absent or unmatched occurrence fails its comparisons. Prose contributes a value occurrence only where the specification requires that result to be stated; an instruction to interpret a result does not by itself require its numerical repetition.

In the running example, if generation must be stated in both a table and the accompanying prose, the two occurrences are assessed separately. A correct table value and an incorrect prose value therefore reduce K relative to reporting both correctly. The score

measures reported calculation results, irrespective of how the system obtained them; semantic interpretation is assessed separately. If no calculated-result occurrences are required, $K = 1$.

Citations and explicit prose requirements (C). We combine citation matching in tables with checks of explicitly declared prose obligations:

$$C = \frac{F_{\text{cit}} + B}{2}, \quad B = \frac{g + v_p + q + a}{4}. \quad (10)$$

F_{cit} applies Equation 2 to citation matches pooled across expected table-cell positions. Let E_x, O_x be the lists of expected and observed citation targets at position x . Targets are compared literally, without source-equivalence rewriting. Define

$$m_{\text{cit}} = \sum_x |\text{set}(E_x) \cap \text{set}(O_x)|, \quad (11)$$

$$n_{e,\text{cit}} = \sum_x |E_x|, \quad n_{o,\text{cit}} = \sum_x |O_x|, \quad (12)$$

$$F_{\text{cit}} = F(m_{\text{cit}}, n_{o,\text{cit}}, n_{e,\text{cit}}). \quad (13)$$

Repeated targets do not create additional matches. Citations in absent expected cells have an empty observed list. Extra-row content is assessed through the table and structure measures.

For prose, g is coverage of required citation groups in their aligned prose components. Each group contains alternative acceptable targets for one obligation. v_p is satisfaction of declared value obligations: numeric obligations check the citation label's value and its unit when an expected unit is specified; unavailable values require their missingness citation, and available textual values require the specified citation. q assesses required record selections, averaging membership F1 and order LCS F1 equally for each selection, using each record's first occurrence to establish order, then averaging across selections. Finally, a is the fraction of observed citation targets allowed in their local prose component.

Let $\mathcal{P}$ be the required prose components, O_p the set of observed targets in component p , and $\mathcal{G}_p$ its indexed collection of required citation groups. Then

$$g = \frac{\sum_{p \in \mathcal{P}} \sum_{\gamma \in \mathcal{G}_p} \mathbf{1}[O_p \cap \gamma \neq \emptyset]}{\sum_{p \in \mathcal{P}} |\mathcal{G}_p|}. \quad (14)$$

For the set $\mathcal{D}$ of declared value requirements, let $I_d = 1$ when requirement d passes the value checks described above, and zero otherwise. Define

$$v_p = \frac{\sum_{d \in \mathcal{D}} I_d}{|\mathcal{D}|}. \quad (15)$$

Both fractions are one when their denominators are zero.

For each record-selection requirement $\ell \in \mathcal{L}$, let E_ℓ, O_ℓ be the expected and observed record sets within its declared record universe, and let $\mathbf{e}_\ell, \mathbf{o}_\ell$ be their corresponding order sequences. The observed order uses each resolved record's first occurrence. Define

$$q_{\ell,\text{set}} = F(|E_\ell \cap O_\ell|, |O_\ell|, |E_\ell|), \quad (16)$$

$$q_{\ell,\text{ord}} = F(\text{LCS}(\mathbf{e}_\ell, \mathbf{o}_\ell), |\mathbf{o}_\ell|, |\mathbf{e}_\ell|), \quad (17)$$

$$q_\ell = \frac{q_{\ell,\text{set}} + q_{\ell,\text{ord}}}{2}, \quad (18)$$

$$q = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} q_\ell. \quad (19)$$

With no record-selection requirements, $q = 1$.

Let L_p be the allowed citation targets for component p , including the permitted missingness forms. Target validity is

$$a = \frac{\sum_{p \in \mathcal{P}} |O_p \cap L_p|}{\sum_{p \in \mathcal{P}} |O_p|}. \quad (20)$$

Observed targets are deduplicated within each component before these counts are pooled. If no citations are observed, a is zero when support groups are required and one otherwise. C assesses declared citations and mechanically checkable obligations; semantic support for their associated claims is assessed separately.

The remaining deterministic measures address missing evidence and maintenance and are reported for their corresponding tasks.

Missingness (M). Let E_m and O_m be the sets of required and emitted unavailable-value citation targets, identified by the missing prefix:

$$M = F(|E_m \cap O_m|, |O_m|, |E_m|). \quad (21)$$

Omissions and spurious markers reduce the score; repeated mentions of the same target do not add credit. When both sets are empty, $M = 1$. Withholding capacity in the running example creates missingness targets for the input and dependent generation result. M assesses their identification; whether the prose explains the analytical limitation is assessed semantically.

Affected-section change rate (A). For maintenance, let $\mathcal{A}$ and $\mathcal{U}$ be the affected and unaffected sections identified during construction. We compare each submitted section with its canonical baseline counterpart after parsing headings, tables, and paragraphs. Line-position metadata is ignored; wording and citation changes count as changes even when meaning is preserved. Let $d_s = 1$ if section s is present and its parsed content differs from the baseline, and zero otherwise. Then

$$A = \frac{1}{|\mathcal{A}|} \sum_{s \in \mathcal{A}} d_s. \quad (22)$$

A is the proportion of affected sections that were changed. Any change receives credit, including a rewording or an incorrect edit; the score does not establish that the required update was completed correctly.

Unaffected-section preservation (P). Let $z_s = 1$ if section s is present and its parsed content equals the baseline, and zero otherwise. Then

$$P = \frac{1}{|\mathcal{U}|} \sum_{s \in \mathcal{U}} z_s. \quad (23)$$

P is the proportion of unaffected sections retained unchanged. Omitted sections receive no credit in either measure. Sections receive equal weight, and each score is one when its respective section set is empty.

Following a capacity update, A records whether the affected sections were changed, while P records whether unrelated sections were preserved. The common measures S, T, K, C, R, J assess the resulting report against the updated requirements, including the correctness of the revised generation value and its interpretation.

A.2 Semantic assessment

Deterministic matching does not establish whether prose satisfies a content requirement or whether cited evidence supports its associated claim. We assess these properties through separate,

evidence-conditioned Boolean judgments. Two section-level assessments address content requirements and citation occurrences respectively, retaining the section’s prose as context. Tables, headings, code blocks, and link-definition metadata are excluded from the candidate text.

Prose requirement coverage (R). The judge receives the section prose, acceptance requirements resolved from the specification, and current reference facts. A requirement may call for an available finding, acknowledgement of missing evidence, or explanation of inapplicability. All applicable clauses must be satisfied for credit; faithful paraphrases, equivalent units, and sensible rounding are accepted. For verdicts $r_{si} \in \{0, 1\}$ over n_s content requirements in section s ,

$$R_s = \frac{1}{n_s} \sum_{i=1}^{n_s} r_{si}, \quad R = \frac{1}{|\mathcal{S}_R|} \sum_{s \in \mathcal{S}_R} R_s, \quad (24)$$

where $\mathcal{S}_R$ contains the required sections with content requirements. Sections receive equal weight, and omitted prose fails its requirements. Separately assessed prohibitions and stylistic constraints do not enter this aggregate.

In the running example, presenting the correct generation value in a table does not satisfy a prose requirement to explain its scenario assumptions. R measures coverage of the specified prose requirements.

Supported citation coverage (J). The second call assesses each citation occurrence against its linked evidence, with the section prose providing context for the associated claim. Support requires consistency with the source’s identity, value, units, scope, and availability. Repeated uses of one source are assessed independently because they may support different claims.

Let $\mathcal{G}_s$ be the indexed collection of required citation groups in section s . For occurrence o , let c_{so} be its citation target and $j_{so} \in \{0, 1\}$ its support verdict. Define

$$b_{sg} = \mathbf{1}[\exists o : c_{so} \in g \text{ and } j_{so} = 1], \quad J = \frac{\sum_s \sum_{g \in \mathcal{G}_s} b_{sg}}{\sum_s |\mathcal{G}_s|}. \quad (25)$$

Each group receives one credit if a supported occurrence cites an acceptable target, and zero otherwise. Omitted groups therefore receive zero. Alternative targets within a group satisfy the same obligation; repeated groups from distinct authored obligations retain their weight. Groups are matched within sections and pooled across the report, giving each required group equal weight. J measures supported coverage of required citation groups, rather than precision over the citations a system chooses to emit.

Invalid judge responses remain failed measurements. With no applicable content requirements or citation groups, the corresponding semantic score is undefined. These assessments concern the represented requirements and linked evidence, rather than exhaustive factuality.

B Model configurations

Table 8 records the configurations used in both generation strategies.

C Estimated cost at current standard rates

Figure 6 compares report quality with estimated generation-response cost at the standard uncached rates listed in Table ?? . Rates were retrieved from OpenRouter on 9 October 2026 for each model author’s standard endpoint.

For report case r generated with model m , let I_r and O_r be its recorded input and output token counts, and let p_m^{in} and p_m^{out} be prices in US dollars per million tokens. We calculate

$$\widehat{\text{cost}}_r = \frac{p_m^{\text{in}} I_r + p_m^{\text{out}} O_r}{10^6}. \quad (26)$$

Every input token is charged at the full input rate, including tokens recorded as cached. No cache, batch, contributor, or promotional discount is applied. Reasoning tokens are already included in the output count.

Estimates cover the returned generation responses retained for all planned cases, including unsuccessful generations.

D Token use

Figures 7–9 compare report quality with input, output, and total token use. Token counts sum the recorded usage of all returned model responses for a report case, including responses from unsuccessful generations and all trajectories under section-based generation. Total tokens equal input plus output tokens.

Each task panel uses all planned cases across 20 projects. Conditions are averaged within projects before projects receive equal weight. The quality coordinate uses $Q = (S + T + K + C + R + J)/6$, with incomplete reports assigned zero quality. Colours identify models and marker shapes distinguish generation strategies, as in the main RQ2 figure.

E Model-response turns

Figure 10 compares report quality with the number of model responses returned during generation. Counts are summed across trajectories, so the section-based values cover all 15 section trajectories. A response can request multiple tool calls; response turns and executed. Failed requests that returned no model response do not contribute a response turn.

F Executed tool calls

Figure 11 compares report quality with executed tool calls recorded in the execution summaries and retained traces. Counts include evidence retrieval, calculation, and terminal submission or preservation tools, and are summed across trajectories.

Table 8: Model configurations. Limits are in tokens. Tool choice refers to ordinary interaction turns; the reserved final turn exposes only terminal submission tools.

Model identifier	Provider / route	Reasoning	Tool choice	Context	Output cap
gpt-5.6-luna	OpenAI direct	Medium	Required	1,050,000	128,000
meta/muse-spark-1.3	OpenRouter / meta	Medium	Auto	1,048,576	943,718
z-ai/glm-5.3-flash	OpenRouter / z-ai/fp8	Low	Auto	1,048,576	131,072
xiaomi/mimo-v2.5	OpenRouter / xiaomi/fp8	Low	Required	1,048,576	131,072
minimax/minimax-m3	OpenRouter / minimax/fp8	Low	Auto	524,288	128,000
mistralai/mistral-small-2603	OpenRouter / mistral	None	Required	262,144	131,072

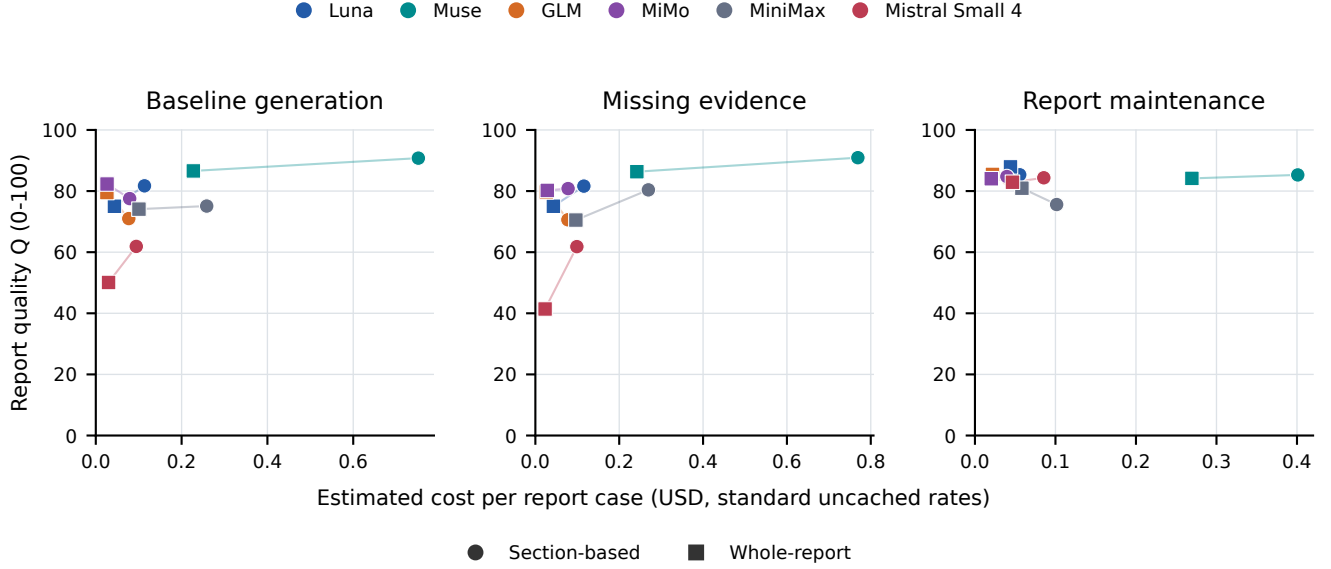


Figure 6: Report quality and estimated generation-response cost at current standard uncached rates. Each panel uses all planned cases across 20 projects, averaging conditions within projects before giving projects equal weight. $Q = (S + T + K + C + R + J)/6$; incomplete reports receive zero quality. Resource totals include returned responses from unsuccessful generations and all trajectories. Colours identify models; circles denote section-based and squares whole-report generation. Lines connect strategies for the same model. Rates are the model-author standard endpoint prices on 9 October 2026, without cache discounts.

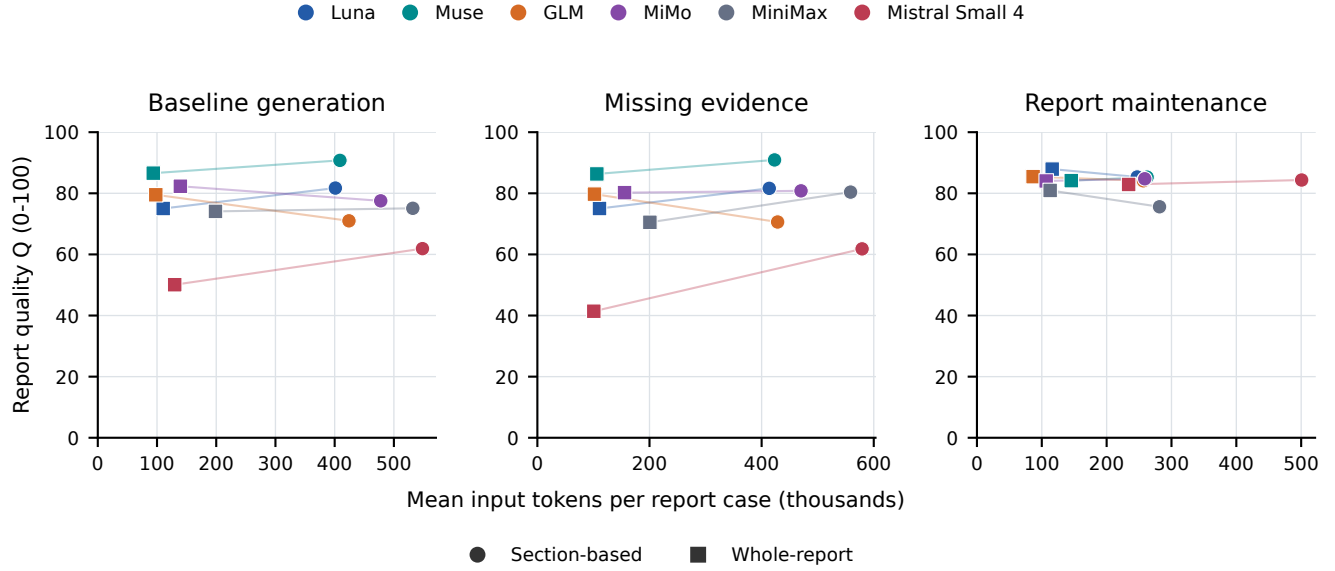


Figure 7: Report quality and input token use. Each panel uses all planned cases across 20 projects, averaging conditions within projects before giving projects equal weight. $Q = (S + T + K + C + R + J)/6$; incomplete reports receive zero quality. Resource totals include returned responses from unsuccessful generations and all trajectories. Colours identify models; circles denote section-based and squares whole-report generation. Lines connect strategies for the same model.

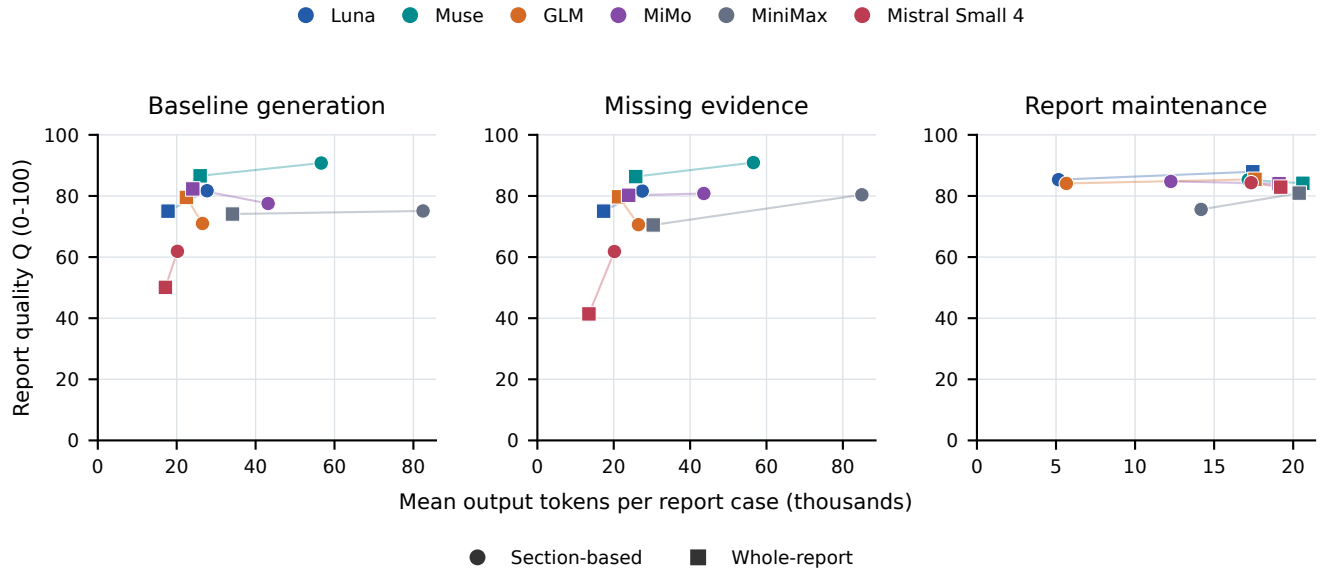


Figure 8: Report quality and output token use. Each panel uses all planned cases across 20 projects, averaging conditions within projects before giving projects equal weight. $Q = (S + T + K + C + R + J)/6$; incomplete reports receive zero quality. Resource totals include returned responses from unsuccessful generations and all trajectories. Colours identify models; circles denote section-based and squares whole-report generation. Lines connect strategies for the same model.

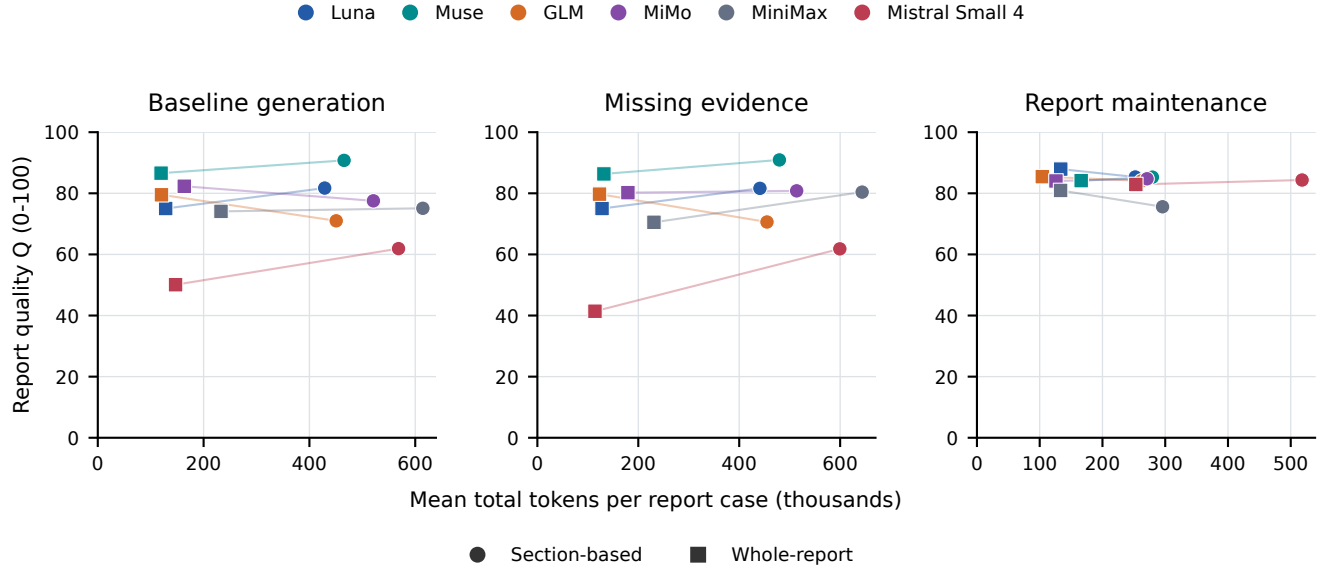


Figure 9: Report quality and total token use. Each panel uses all planned cases across 20 projects, averaging conditions within projects before giving projects equal weight. $Q = (S + T + K + C + R + J)/6$; incomplete reports receive zero quality. Resource totals include returned responses from unsuccessful generations and all trajectories. Colours identify models; circles denote section-based and squares whole-report generation. Lines connect strategies for the same model.

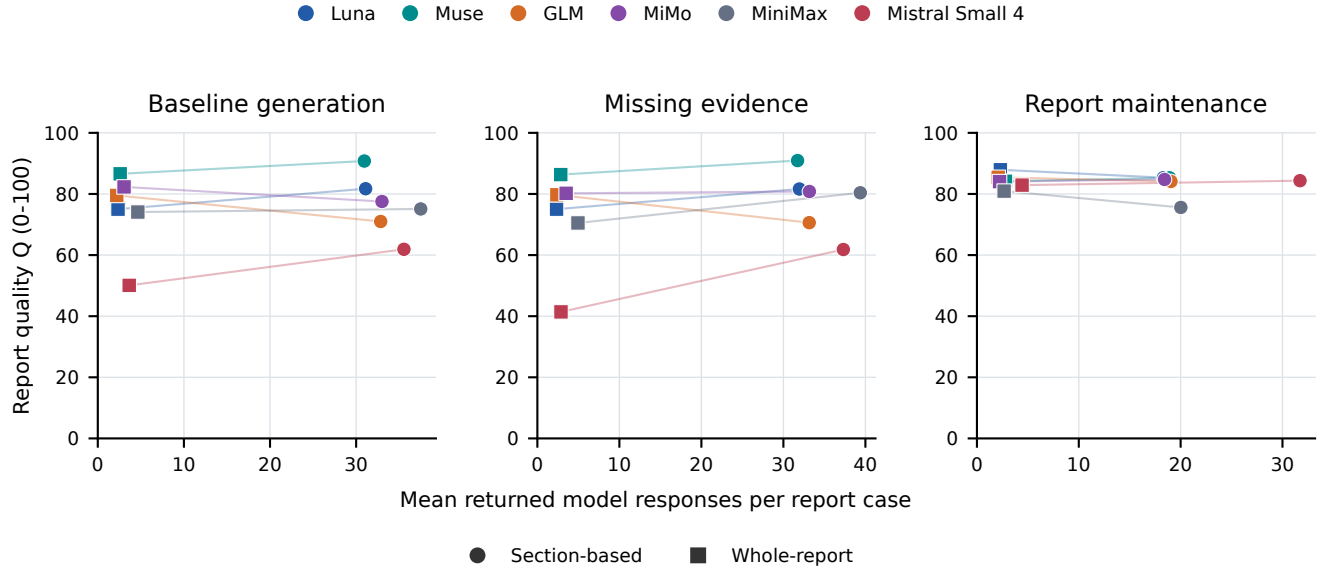


Figure 10: Report quality and returned model-response turns. Each panel uses all planned cases across 20 projects, averaging conditions within projects before giving projects equal weight. $Q = (S + T + K + C + R + J)/6$; incomplete reports receive zero quality. Resource totals include returned responses from unsuccessful generations and all trajectories. Colours identify models; circles denote section-based and squares whole-report generation. Lines connect strategies for the same model.

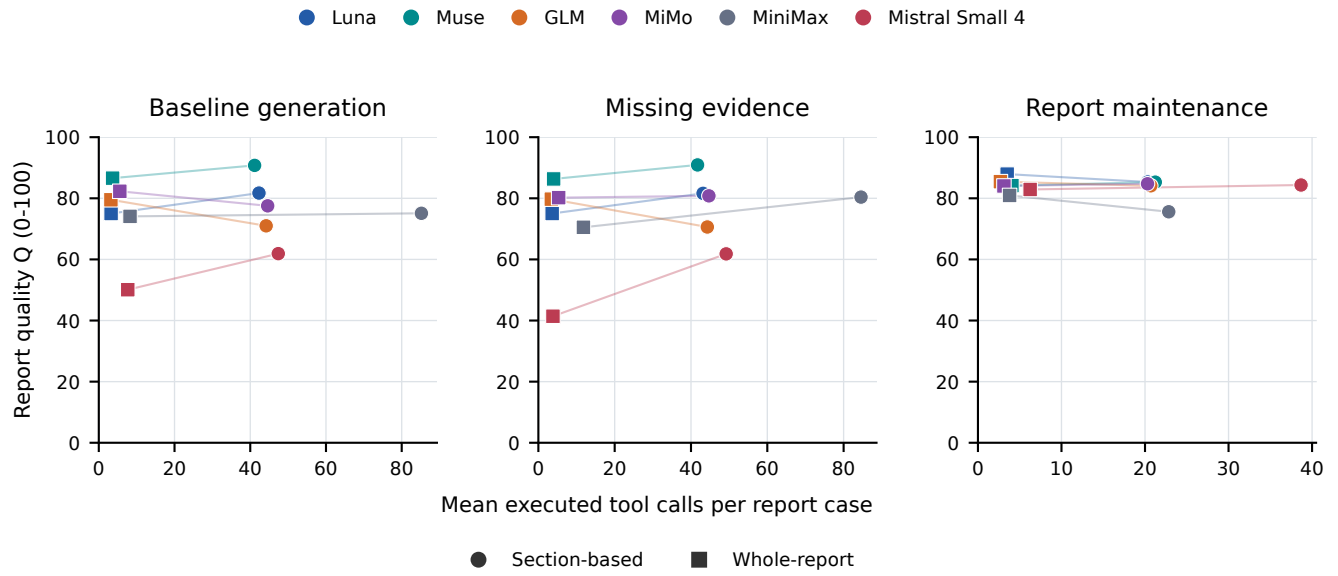


Figure 11: Report quality and mean executed tool calls, including terminal submission or preservation tools. Both coordinates use all planned cases and the same project aggregation as the main analysis. Tool counts use native execution summaries supplemented by execution-observation and attempt-ledger reconstruction for 15 MiniMax whole-report cases. Replayed cached actions are not counted again. Colours identify models; circles denote section-based and squares whole-report generation. Lines connect strategies for the same model.

