

Investigating Scoping Errors in Open-Domain Numerical Reasoning

MSc Thesis in Data Science & AI Technology
 Moniek Smink

$$\nabla \cdot \mathbf{v} = 0$$

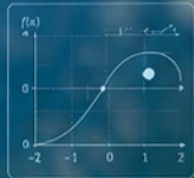
$$\frac{\partial u}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla \pi + \nu \nabla^2 \mathbf{u} + \mathbf{S}$$

$$\frac{\partial \pi}{\partial t} + \mathbf{u} \cdot \nabla \pi = -D \nabla^2 \pi + S$$

$$\partial T + \mathbf{u} \cdot \nabla T = \kappa \nabla^2 T + Q$$

$$-\frac{\hbar^2}{2m} \nabla^2 \psi + V\psi = i\hbar \frac{\partial \psi}{\partial t}$$

$$\frac{d}{dt} E(t) = -\lambda E(t) + u(t)$$



$$y = a_0 + \sum_{n=1}^N \left(a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right)$$

$$a_0 = \frac{2}{L} \int_0^L y(x) \cos \frac{n\pi x}{L} dx$$

$$b_n = \frac{2}{L} \int_0^L y(x) \sin \frac{n\pi x}{L} dx$$

Parameters:
 $\mu = 0.012$
 $\sigma = 1.25$
 $\rho = 0.87$
 $\Delta t = 0.01$
 $\epsilon = 1.0 \times 10^{-6}$



Boundary conditions:
 $u(x, 0) = 0, \quad x \in \Omega$
 $u(\partial D, t) = 0, \quad t > 0$
 $\frac{\partial u}{\partial n}(\partial D, t) = g(t), \quad t > 0$

Algorithm:
 1. Initialize $x_0 \in \mathbb{R}^n$.
 2. For $k = 0, 1, 2, \dots$
 $x_{k+1} = x_k - \alpha \nabla f(x_k)$
 3. Until $\|\nabla f(x_k)\| < \epsilon$
 4. Return $x^* = x_k$

Model:

$$x_{k+1} = Ax_k + Bu_k$$

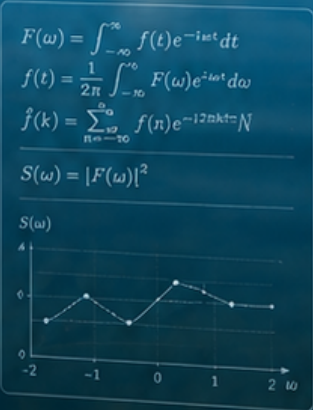
$$y_k = Cx_k + Du_k$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

$$P(X \leq x) = \int_{-\infty}^x f_X(t) dt$$

$$\min_x f(x) \quad \text{s.t. } g_1(x) \leq 0, h_1(x) = 0$$



Investigating Scoping Errors in Open-Domain Numerical Reasoning

by

Moniek Smink

to obtain the degree of

Master of Science

in

Data Science & Artificial Intelligence Technology

at the Delft University of Technology,

to be defended publicly on Wednesday June 24th, 2026 at 9:30 AM.

Student number: 6167667
Project duration: November 3rd, 2025 – June 24th, 2026
Thesis committee: Prof. dr. ir. A. Anand, TU Delft, supervisor
Dr. E. S. Verwer, TU Delft

Web Information Systems Group

Faculty EEMCS, Delft University of Technology

Delft, the Netherlands

www.ewi.tudelft.nl

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Preface

This thesis marks the culmination of my Master's journey in Data Science and Artificial Intelligence Technology at TU Delft. It has been both a technical undertaking and an intellectual exploration into the reasoning failures of large language models (LLMs), particularly in critical numerically grounded settings. Through this work, I learned not only how to build and evaluate complex experimental pipelines, but also how to approach research with depth, patience, and independence.

I would like to sincerely thank my supervisors, Dr. Avishek Anand and Dr. Venkatesh Viswanathan, for their guidance, feedback, and support throughout this process. Their insights helped shape the direction of this thesis, and their encouragement allowed me to work independently while still feeling supported when the process became unclear. I would also like to thank my fellow Master's student peers at the weekly research group whose enthusiasm and kindness made each week warm and something to look forward to.

I am grateful to TU Delft for providing the academic environment, resources, and freedom to pursue independent research. The university has given me the space to think critically, develop my own ideas, and grow both as a researcher and a person.

I am also profoundly thankful to my family, especially my mother, father, brother, and uncle, for their constant love, patience, and belief in me. Their support has carried me through far more than just this thesis, and I am so incredibly happy to share this process with them. To the friends I have made here, thank you for making Delft feel like home, for the deep conversations, laughter, and reminders that life exists beyond lecture rooms and deadlines.

Finally, a small note of appreciation to my instruments, which gave me moments of quiet reflection throughout this journey. In between long days of writing, coding, and thinking, they offered space to pause, reset, and remember why creativity and passion matters.

*Moniek Smink
Delft, June 2026*

Abstract

Numerical information shapes how people interpret real-world events, evaluate claims, and make decisions. As Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) systems are increasingly used for open-domain information seeking, their ability to reason reliably over quantities has become critical. Yet, even when models retrieve relevant evidence and produce fluent chain-of-thought reasoning, they still often reach incorrect conclusions by failing to reason about numbers in the right contexts. This work investigates *numerical scoping errors*, an underexplored reasoning failure where a model incorrectly binds a quantity to the wrong referent, time-frame, unit, population, condition, or source. In high impact settings, a single mis-scoped number can change the meaning of a conclusion, turning a statistic into a misleading financial signal, an unsafe medical recommendation, or a false public claim. This makes numerical scoping errors an important reasoning failure to investigate. When performing numerical fact-checking (FC) and question-answering (QA), this work identifies scoping errors as a common and consequential failure mode in generated LLM reasoning traces across tasks. In human analysis, scoping errors appeared in 44.7% of numerical FC and 26.8% of numerical QA traces, with 35.3% and 81.8% of these errors, respectively, judged to contribute to incorrect final answers. To detect these failures at scale, this work develops a hybrid LLM-as-a-Judge pipeline that decomposes reasoning traces into quantities, verifies whether each quantity is correctly scoped based on its source type, and aggregates these quantity-level judgements into trace-level labels. The pipeline achieves moderate agreement with human annotations, but can be expensive to leverage. To explore more efficient alternatives, the hybrid pipeline’s scoping detection signal is distilled into smaller, conservative student models which process traces around 212 times faster. Finally, this work evaluates whether scoping-aware reward signals can improve downstream performance in a parallel test-time scaling setting, where the system chooses among multiple generated reasoning traces, producing modest improvements, especially when combined with correctness-based selection. Overall, this work shows that numerical reasoning failures are not merely errors of calculation but errors of context. By making these contextual failures visible and measurable, this work takes a step toward numerical reasoning systems that can support high-impact, real-world decisions not only because they calculate correctly, but because they understand what their numbers mean.

Contents

Preface	i
Abstract	ii
1 Introduction	1
1.1 Background & Motivation	1
1.2 Research Questions	4
1.3 Scientific Contributions	4
1.4 Thesis Outline	4
2 Related Work	5
2.1 Chain-of-Thought in LLMs	5
2.2 Open-Domain Numerical Reasoning Tasks	5
2.3 Reasoning Failure Typologies in LLMs	6
2.3.1 Scoping Errors	6
2.4 Reasoning Failure Measurement Strategies in LLMs	6
2.5 Mitigation of Reasoning Failures in LLMs	7
2.5.1 Test-Time Scaling and Reward-Guided Selection	7
2.6 Position of This Work	8
3 Methodology	9
3.1 Inference Pipelines	9
3.1.1 Veracity Prediction	9
3.1.2 Numerical Question Answering	10
3.2 Reasoning Failure Typology for Numerical Reasoning in LLMs	10
3.2.1 Intuitive & Logical Reasoning Failures	10
3.2.2 Numerical Scoping Errors as Representation Failures	11
3.2.3 Relevance of Typology & Scoping Error Characterization	12
3.3 Measuring Numerical Scoping Errors	13
3.3.1 Human Qualitative Analysis	13
3.3.2 Straight LLM-as-a-Judge	13
3.3.3 Hybrid LLM-as-a-Judge Scoping Failure Detection Pipeline	14
3.3.4 Distilling the Hybrid LLM-as-a-Judge Pipeline into a Student Model	16
3.4 Mitigating Scoping Errors with Parallel Test-Time Scaling	17
3.4.1 Candidate Trace Generation	17
3.4.2 Scoping Error Filtering with the Hybrid LLM-as-a-Judge Pipeline	17
3.4.3 Learned Scoping-Aware Reward Models	17
3.4.4 Overview of Scoping-Aware Mitigation Variants	18
4 Implementation Details & Experiments	19
4.1 Experimental Overview	19
4.2 Datasets & Evidence Configuration	19
4.2.1 QUANTEMP	19
4.2.2 FinQA	20
4.3 Model & Generation Setup	20
4.3.1 Inference Model	20
4.3.2 Judge Model	20
4.3.3 Reward Models	21
4.3.4 Prompt Templates & Output Formats	21
4.4 Scoping Error Measurement Implementation	21
4.4.1 Human Annotation Setup	21

4.4.2	Straight LLM-as-a-Judge Setup	22
4.4.3	Hybrid Judge Pipeline Implementation	22
4.4.4	Distilled Scoping Measurement	23
4.5	Reward Model Training	23
4.5.1	Training Data Construction	23
4.5.2	Training Procedure	24
4.6	Parallel Test-Time Scaling Experiments	25
4.6.1	Test-Time Candidate Trace Generation	25
4.6.2	Baseline Scoping-Blind Aggregation Methods	25
4.6.3	Scoping-Aware Test-Time Scaling Variants	26
4.6.4	Overview of Mitigation Variants	27
4.6.5	Test-Time Scaling Fallback Rules	27
4.7	Evaluation Metrics	28
4.7.1	Downstream Outcome-Level Metrics	28
4.7.2	Reasoning Failure Prevalence & Pestilence	28
4.7.3	Human Annotator, LLM Judge, & Distilled Student Agreement Metrics	28
4.7.4	Invalidity, Diversity & Runtime Metrics	29
4.7.5	Statistical Testing	29
4.8	Hardware & Software Configuration	29
5	Results	30
5.1	RQ1: Identifying A Common Reasoning Failure in RAG-based Numerical Reasoning	30
5.1.1	Generating Reasoning Traces Across Tasks	30
5.1.2	Prevalence & Pestilence of Common Reasoning Failures	31
5.1.3	Taxonomizing Numerical Scoping Failures	33
5.2	RQ2: Measuring Numerical Scoping Failures in RAG-based Numerical Reasoning	34
5.2.1	Distilling the Hybrid LLM-as-a-Judge Pipeline Signal	35
5.3	RQ3: Mitigating Numerical Scoping Failures & Effects on Downstream Performance	36
5.3.1	Candidate Trace Generation	36
5.3.2	Impacts of Test-Time Scaling on Downstream Performance	37
5.3.3	Scoping-Aware Mitigation Overview	42
6	Conclusion	44
6.1	Limitations	45
6.2	Future Work	45
6.3	Disclosure	45
	References	46
A	Appendix A	50
A.1	Human Qualitative Analysis Guidelines	50
A.1.1	Numerical Scoping Errors	50
A.1.2	Arithmetic Errors	50
A.1.3	Looping Errors	51
A.1.4	Annotation Procedure	51
A.2	Mathematical Formulae Required for Methodology	51
A.2.1	Cosine Similarity	51
A.2.2	F1 Score & Averaging Methods	51
A.2.3	ROC-AUC & Label Agreement	52
A.2.4	Answer Entropy & Disagreement	53
A.3	LLM Prompts	53
B	Appendix B	61
B.1	Training of Learned ORMs for Test-Time Scaling Experiments	61
B.2	Scoping Failure Detection Examples	62
B.3	Scoping-Aware Mitigation Examples	65
B.4	Ablations	67
B.4.1	Generation Model Ablation	67
B.4.2	Verifier Ablations	68

List of Figures

1.1	Examples of reasoning failures in LLM reasoning traces on open-domain numerical tasks, demonstrating the diversity of failure modes: (top left) a logical error where an incorrect magnitude comparison is drawn despite correct value retrieval; (top right) a parametric memory error, where an incorrect formula is used, and an arithmetic error, where the ensuing calculation is performed incorrectly; (bottom) an answer inconsistency error where a conclusion contradicts correct intermediate reasoning.	2
1.2	Numerical scoping error examples where a reasoning model (Deepseek-DistillQwen-7B; DeepSeek-AI et al. [13]) mis-scopes real-world statistics in the QUANTEMP numerical fact-checking dataset [53] and FinQA numerical question-answering (QA) dataset [9].	3
3.1	Overview of the inference pipelines used for veracity prediction (green) and numerical question answering (orange). In both settings, task inputs, few-shot examples, and evidence are provided to an inference model, which generates an intermediate reasoning trace, final prediction, and justification.	9
3.2	Overview of the reasoning failure typology, refined from [48, 46, 22, 5, 2, 15], used to contextualize numerical scoping errors. Failures are organized into intuitive and logical branches, with scoping errors positioned primarily as logical failures that can arise during problem representation or problem execution. Arithmetic and looping errors are shown as related but distinct failure modes.	10
3.3	Overview of the hybrid LLM-as-a-Judge pipeline for measuring numerical scoping errors. The pipeline extracts statistics from the task inputs and reasoning trace, classifies trace statistics as extractive, extrapolative, or parametric, verifies their scope using type-specific judge calls, and combines statistic-level scoping judgements with normalized importance weights to produce a trace-level scoping error label.	14
3.4	Example of scoping detection by hybrid LLM-as-a-Judge pipeline with a numerical QA query from FinQA [9]. The pipeline extracts statistics from the task inputs x , E and reasoning trace R , classifies trace statistics Q_R as extractive, extrapolative, or parametric, verifies their scope using type-specific judge calls, and combines statistic-level scoping judgements with normalized importance weights to produce a <i>True</i> trace-level scoping error label.	16
4.1	Overview of the scoping-aware test-time scaling variants used to evaluate numerical scoping error mitigation.	27
5.1	Examples of arithmetic and looping reasoning failures in LRM reasoning traces on numerical tasks.	31
5.2	Downstream performance of scoping-aware and scoping-blind test-time scaling strategies across different numbers of candidate traces (K) from the Deepseek-Distill-Qwen-7B (left) and Deepseek-Distill-Llama-8B (right) inference models. Scoping-aware strategies include the Hybrid LLM-as-a-Judge Pipeline (HP), Scoping Outcome Reward Model (S-ORM), and Multi-Task ORM (MT-ORM). Scoping-blind strategies include Self-Consistency and the Correctness ORM (C-ORM). For QUANTEMP, performance is reported as weighted-F1; for FinQA, performance is reported as exact match accuracy.	39
5.3	Marginalized and conditional probabilities of the co-occurrence of correctness and scoping errors across different numbers of Distill-Qwen-generated candidate traces (K) for QUANTEMP and FinQA test sets. Clustered logistic regression based on generalized estimating equations, clustered on query identity was used to determine statistical significance at a confidence level of 0.95. A statistical significance level of $p < 0.05$ is marked with *, $p < 0.01$ with **, and $p < 0.001$ with ***.	41

5.4	Average correctness outcome reward model (ORM) scores given the QUANTEMP or FinQA Distill-Qwen-generated trace did or did not have a detected scoping error according to the Hybrid LLM-as-a-Judge pipeline, across different numbers of candidate traces (K). Clustered linear regression based on generalized estimating equations, clustered on query identity was used to determine statistical significance at a confidence level of 0.95. A statistical significance level of $p < 0.05$ is marked with *, $p < 0.01$ with **, and $p < 0.001$ with ***.	42
A.1	Structure of the numerical fact-checking prompt.	54
A.2	Structure of the numerical QA prompt.	55
A.3	Structure of the Straight LLM-as-a-Judge numerical fact-checking prompt.	56
A.4	Structure of the Straight LLM-as-a-Judge numerical question-answering prompt.	57
A.5	Structure of the extractive statistic scoping verification prompt used in the rubric-based LLM-as-a-Judge pipeline. Task-specific differences are highlighted in blue for fact-checking and red for numerical question answering.	58
A.6	Structure of the non-extractive statistic scoping verification prompt used in the rubric-based LLM-as-a-Judge pipeline. The prompt classifies non-extractive statistics as extrapolative or parametric and verifies their correctness or plausibility. Task-specific differences are highlighted in blue for fact-checking and red for numerical question answering.	59
A.7	Structure of the statistic importance weight judging prompt used in the rubric-based LLM-as-a-Judge pipeline. The prompt assigns normalized statistic importance weights based on each statistic's causal influence on the final answer and justification. Task-specific differences are highlighted in blue for fact-checking and red for numerical question answering.	60
A.8	Structure of the outcome reward model prompts for fact-checking (blue) and numerical question answering (red).	60
B.1	Training learning curves for the learned outcome reward models (ORMs) used in test-time scaling experiments. The figures show training and validation accuracies and training loss for the scoping-blind correctness ORM, single-task scoping ORM, and multi-task scoping & correctness ORM. For the multi-task ORM, training loss is the total of both the correctness and scoping losses and accuracies are the unweighted average of the correctness and scoping accuracies.	61
B.2	Downstream performance of the Multi-Task ORM's Best-of-N using combined scoping and correctness scores, aggregated with a scoping weight of α . Downstream performance is the weighted-F1 and exact match accuracy calculated after test-time scaling with $K = 12$ or $K = 20$ candidate traces for the QUANTEMP and FinQA datasets, respectively. The selected α of 0.18 is marked with a blue dotted line. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct.	69
B.3	Structure of the evidence-aware outcome reward model prompts for fact-checking.	69

List of Tables

1.1	Examples of complex open-domain numerical reasoning in fact-checking (FC) and question-answering (QA): (top) an FC claim whose evidence requires reasoning about and understanding a multi-aspect, domain-specific tax system proposed in American politics to generate a nuanced label; (middle) an FC claim requiring multiple computational steps and temporal contextualization to arrive at the final label; (bottom) a QA query which requires combining understanding of domain-specific taxation terms in a time period in both tabular and textual formats to arrive at a numerical answer.	1
2.1	Comparison of this thesis with representative related work across task setting, failure analysis, automatic detection, and mitigation dimensions.	8
3.1	Examples of numerical scoping error types identified in model reasoning traces for numerical fact-checking and question-answering tasks. Mis-scoped numerical quantities are highlighted in red.	12
4.1	Distribution of reward model training data after trace generation, de-duplication, and stratified sampling based on correctness labels.	24
4.2	Overview of test-time scaling variants used to evaluate numerical scoping error mitigation.	27
5.1	Fact-checking outcome-level performance across claim taxonomy categories, verdict categories, and evidence classes on the QUANTEMP test set. Occasionally, reasoning trace outcomes were invalid (0.4-2.2% of traces), resulting in exclusion.	30
5.2	Question-answering outcome-level performance across evidence classes on the FinQA test set. Reasoning trace outcomes were sometimes invalid (22.7-27.3% of traces), resulting in exclusion.	31
5.3	Reasoning failure prevalence as judged by a human annotator on 79 random reasoning traces generated by Deepseek-Distill-Qwen-7B on 38 fact-checking and 41 question-answering queries.	32
5.4	Reasoning failure prevalence as judged by a straight LLM-as-a-Judge (Qwen2.5-14B) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B. Occasionally, reasoning trace outcomes were invalid (<0.25% of traces), resulting in exclusion.	32
5.5	Manual evaluation of straight LLM-as-a-Judge by a human annotator on stratified samples of fact-checking (n=38) and question-answering (n=41) reasoning traces generated by Deepseek-Distill-Qwen-7B and judgements generated by Qwen2.5-14B-Instruct.	33
5.6	Scoping failure prevalence at a trace and statistic-level as judged by a hybrid LLM-as-a-Judge pipeline (Qwen2.5-14B) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B with different evidence splits. Rarely, judge outcomes were invalid (<0.8% of traces), resulting in exclusion.	34
5.7	Statistic-level failure prevalence for extractive, extrapolative, and parametric scoping errors as judged by a hybrid LLM-as-a-Judge pipeline (Qwen2.5-14B) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B with different evidence splits. Rarely, judge outcomes were invalid (<0.8% of traces), resulting in exclusion.	35
5.8	Manual evaluation of hybrid LLM-as-a-Judge scoping error detection pipeline by a human annotator on stratified samples of fact-checking (n=38) and question-answering (n=41) reasoning traces generated by Deepseek-Distill-Qwen-7B and judgements generated by Qwen2.5-14B-Instruct.	35

5.9	Evaluation of student models taught by the hybrid LLM-as-a-Judge scoping detection pipeline. Agreement is calculated between teacher scoping-error labels for whole test sets of QUANTEMP and FinQA and stratified samples of fact-checking ($n=38$) and question-answering ($n=41$) labels annotated by a human. Relative speed is calculated on average between the student and teacher models on the whole test sets. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct. The correctness head of the multi-task ORM is not evaluated.	36
5.10	Average fact-checking outcome-level performance across generation temperatures for $K = 12$ candidate traces ($k = 3$ per temperature) on the QUANTEMP test set ($n=2,495$) with <i>Gold & Re-Ranked</i> evidence. Traces were generated from either Deepseek-Distill-Qwen-7B or Deepseek-Distill-Llama-8B. Reasoning trace outcomes were sometimes invalid (0.4-0.6% of Distill-Qwen traces; 0.01-0.1% of Distill-Llama traces), resulting in exclusion.	37
5.11	Average question-answering outcome-level performance across generation temperatures for $K = 20$ candidate traces ($k = 5$ per temperature) on the FinQA test set ($n=1,147$) with <i>Whole-Document</i> evidence. Traces were generated from either Deepseek-Distill-Qwen-7B or Deepseek-Distill-Llama-8B. Reasoning trace outcomes were sometimes invalid (24.9-28.5% of Distill-Qwen traces; 7.1-8.2% of Distill-Llama traces), resulting in exclusion.	37
5.12	Downstream performance of different scoping error-aware test-time scaling strategies for $K = 12$ fact-checking candidate traces and $K = 20$ numerical QA candidate traces on the QUANTEMP ($n=2,495$) and FinQA ($n=1,147$) test sets. Traces were generated from either Deepseek-Distill-Qwen-7B or Deepseek-Distill-Llama-8B. Reasoning trace outcomes were sometimes invalid (0.0-2.1% of traces), resulting in exclusion.	38
5.13	Average answer entropy and disagreement across varying K values for QUANTEMP ($n = 2,495$) and FinQA ($n = 1,147$) test set Distill-Qwen-generated candidate traces with and without scoping errors. Reasoning trace outcomes were sometimes invalid (0.4-8.1% of traces), resulting in exclusion. Due to computational limits, QUANTEMP test-time scaling experiments were only run with a maximum K value of 12.	41
B.1	Example of numerical scoping failure detection by the Hybrid LLM-as-a-Judge Pipeline on a FinQA <i>Whole-Document</i> LLM reasoning trace. Mistakes in scoping failure detection are in red. Statistics are colored by type.	62
B.2	Another example of numerical scoping failure detection by the Hybrid LLM-as-a-Judge Pipeline on a FinQA <i>Whole-Document</i> LLM reasoning trace. Mistakes in scoping failure detection are in red. Statistics are colored by type.	63
B.3	Example of numerical scoping failure detection by the Hybrid LLM-as-a-Judge Pipeline on a QUANTEMP <i>Re-Ranked</i> LLM reasoning trace. Mistakes in scoping failure detection are in red. Statistics are colored by type.	64
B.4	Another example of numerical scoping failure detection by the Hybrid LLM-as-a-Judge Pipeline on a QUANTEMP <i>Re-Ranked</i> LLM reasoning trace. Mistakes in scoping failure detection are in red. Statistics are colored by type.	64
B.5	Example of LLM candidate traces chosen by the various test-time scaling variants on a QUANTEMP claim with <i>Re-Ranked</i> evidence. The Correctness Outcome Reward Model Best-of-N (C-ORM: BoN) variant chooses a trace based on a quality score; the Hybrid Pipeline Scoping Filtering + C-ORM BoN (HP: SF + C-ORM: BoN) variant chooses a trace based on first filtering for non-scoping failure traces and then choosing based on a quality score; the Scoping-ORM Scoping Filtering + C-ORM BoN (S-ORM: SF + C-ORM: BoN) variant filters traces based on a distilled scoping error signal, then chooses based on the C-ORM quality score; the Multi-Task ORM Scoping Filtering + BoN (MT-ORM: SF + BoN) variant filters traces based on a distilled scoping error signal, then chooses based on the MT-ORM's quality score. Finally, the Multi-Task ORM Scoping & Correctness Best-of-N (MT-ORM: BoN) chooses a trace based on a combined scoping failure and correctness score. Scoping failures are highlighted in red. Verdicts are colored by correctness.	65

B.6	Example of LLM candidate traces chosen by the various test-time scaling variants on a <i>FinQA</i> question with <i>Whole-Document</i> evidence. The Correctness Outcome Reward Model Best-of-N (C-ORM: BoN) variant chooses a trace based on a quality score; the Hybrid Pipeline Scoping Filtering + C-ORM BoN (HP: SF + C-ORM: BoN) variant chooses a trace based on first filtering for non-scoping failure traces and then choosing based on a quality score; the Scoping-ORM Scoping Filtering + C-ORM BoN (S-ORM: SF + C-ORM: BoN) variant filters traces based on a distilled scoping error signal, then chooses based on the C-ORM quality score; the Multi-Task ORM Scoping Filtering + BoN (MT-ORM: SF + BoN) variant filters traces based on a distilled scoping error signal, then chooses based on the MT-ORM's quality score. Finally, the Multi-Task ORM Scoping & Correctness Best-of-N (MT-ORM: BoN) chooses a trace based on a combined scoping failure and correctness score. Scoping failures are highlighted in red. Verdicts are colored by correctness.	66
B.7	Scoping failure prevalence at a trace and statistic-level as judged by a hybrid LLM-as-a-Judge pipeline (Qwen2.5-14B-Instruct) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B and Deepseek-Distill-Llama-8B with different evidence splits. Rarely, judge outcomes were invalid (<0.8% of traces), resulting in exclusion.	67
B.8	Statistic-level failure prevalence for extractive, extrapolative, and parametric scoping errors as judged by a hybrid LLM-as-a-Judge pipeline (Qwen2.5-14B) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B and Deepseek-Distill-Llama-8B with different evidence splits. Rarely, judge outcomes were invalid (<0.8% of traces), resulting in exclusion.	68
B.9	Evaluation of student models taught by the hybrid LLM-as-a-Judge scoping detection pipeline with different threshold values (λ) for when to consider a trace as having a scoping error. Agreement is calculated between teacher scoping error labels for the whole test sets of QUANTEMP and <i>FinQA</i> . The weighted-F1 and exact match accuracy with ground truth labels are calculated after test-time scaling with $K = 12$ or $K = 20$ candidate traces for QUANTEMP and <i>FinQA</i> , respectively through test-time scaling. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct. The correctness head of the multi-task ORM is not evaluated.	68
B.10	Evaluation of evidence-aware student models taught by the hybrid LLM-as-a-Judge scoping detection pipeline. Agreement is calculated between teacher scoping error labels for whole test set of QUANTEMP and stratified samples of fact-checking (n=38) labels annotated by a human. Relative speed is calculated on average between the student and teacher models on the whole test sets. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct. The correctness head of the multi-task ORM is not evaluated.	69
B.11	Downstream performance of different evidence-aware scoping error-aware test-time scaling strategies for $K = 12$ fact-checking candidate traces on the QUANTEMP (n=2,495) test set. Reasoning trace outcomes were sometimes invalid (0.0-2.1% of traces), resulting in exclusion.	70
B.12	Evaluation of student multi-task ORMs taught by the hybrid LLM-as-a-Judge scoping detection pipeline, using unweighted task loss aggregation or Soft Optimal Uncertainty Weighting (UW-SO). Agreement is calculated between teacher scoping error labels for whole test sets of QUANTEMP and <i>FinQA</i> and stratified samples of fact-checking (n=38) & numerical QA (n=41) labels annotated by a human. Relative speed is calculated on average between the student and teacher models on the whole test sets. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct. The correctness head of the multi-task ORM is not evaluated.	70
B.13	Downstream performance of different scoping-aware mitigation variants using Multi-Task ORMs trained with or without Soft Optimal Uncertainty Weighting (UW-SO) [27]. Mitigation variants used $K = 12$ or $K = 20$ candidate reasoning traces from the QUANTEMP (n=2,495) or <i>FinQA</i> (n=1,147) test sets.	71

1

Introduction

1.1. Background & Motivation

Numerical information is central to how people interpret real-world events, make decisions, and evaluate claims [43]. News articles, public-health reports, financial documents, scientific summaries, and policy debates frequently rely on quantities such as percentages, rates, dates, measurements, and comparisons to state their case. Errors in these quantities can therefore have practical consequences: a misread political statistic can change votes, a badly contextualized financial figure can influence investors, and an incorrect medical dosage can affect someone's health.

This makes mixed numerical reasoning an important capability for language technologies. Unlike isolated arithmetic problems, real-world numerical reasoning often requires models to combine natural-language understanding with numerical understanding: identifying the relevant quantity, determining what entity or condition it refers to, comparing it to other quantities, and sometimes performing calculations over evidence. Table 1.1 depicts such examples of complex open-domain numerical reasoning and shows that such problems are multi-aspect: requiring multiple numbers and reasoning steps, often domain-specific: concerning particular fields such as taxation and politics, and nuanced: with labels and patterns that are not always deterministic. Existing benchmarks reflect this diversity and show that numerical reasoning is not a narrow mathematical skill, but a recurring requirement across many information-seeking tasks [53, 31, 9, 37].

Table 1.1: Examples of complex open-domain numerical reasoning in fact-checking (FC) and question-answering (QA): (top) an FC claim whose evidence requires reasoning about and understanding a multi-aspect, domain-specific tax system proposed in American politics to generate a nuanced label; (middle) an FC claim requiring multiple computational steps and temporal contextualization to arrive at the final label; (bottom) a QA query which requires combining understanding of domain-specific taxation terms in a time period in both tabular and textual formats to arrive at a numerical answer.

Task	Instance
FC	<p>Claim: "The expanded child tax credit in the Build Back Better Act has "no means testing," so people making \$200,000 and \$400,000 would get "the same as someone making" \$70,000."</p> <p>Evidences:</p> <ol style="list-style-type: none">1. "while most democratic senators are in favor of the expanded child tax credit, uncertainty surrounding the build back better act is putting the potential of an extension in doubt..."2. "the increased child tax credit is reduced by \$50 for every \$1,000 income above the thresholds, for instance, if you are filing for a single return and your annual income is \$77,000, you will only be eligible for \$3,500 for each child under 6 years old and \$2,900 for each child under 18 years old, this reduction only works on the increased tax credit amount, you will still be eligible for at least \$2,000 tax credit for every dependent below the age of 18 years old, so long as your annual income is less than \$200,000 for individuals and \$400,000 for married couples." <p>Label: Conflicting</p>
FC	<p>Claim: Nearly 1,000 people were shot and killed by police in the U.S. between April 2020 and April 2021."</p> <p>Evidences:</p> <ol style="list-style-type: none">1. "seventy-three u.s. police officers were gunned down in 2021, during that same year, officers shot and killed roughly sixteen times more civilians. All countries..."2. "reporting that 1,021 people were shot and killed by police in 2020, the small number of people at risk for violence by helping them" <p>Label: Supports</p>
QA	<p>Question: "As of December 31, 2008, what percentage of unrecognized tax benefits would not impact the effective if recognized?"</p> <p>Evidences:</p> <ol style="list-style-type: none">1. "notes to the consolidated financial statements the activity in the accrued liability for unrecognized tax benefits for the two years ended December 31, 2008 was as follows: (millions) 2008 2007 balance at January 1 \$ 110 \$ 77 additions based on tax positions related to the current year 12 21 additions for tax positions of prior years 5 19 reductions for tax positions of prior years -17 (17) -5 (5) pre-acquisition unrecognized tax benefits 20 2014 reductions for expiration of the applicable statute of limitations -6 (6) -5 (5) settlements -21 (21) -1 (1) currency -4 (4) 4 balance at December 31 \$ 99 \$ 110"2. "if recognized, \$ 89 million and \$ 88 million would impact the effective rate as of December 31, 2008 and 2007, respectively." <p>Answer: 10%</p>

The importance of open-domain numerical reasoning has become more pronounced with the rise of LLM-based assistants and search systems. Users increasingly ask models questions that require both retrieval and reasoning over diverse, real-world numerical evidence. Prior work shows that open-domain numerical reasoning benchmarks remain challenging for neural and Large Language Models (LLMs); for example Li et al. [31] reports that state-of-the-art models remain substantially below human performance across arithmetic reasoning tasks. Improving numerical reasoning is necessary not only for benchmark performance, but for the reliability of LLMs in high-impact real-world settings.

Chain-of-thought (CoT) prompting has substantially improved the performance of LLMs on complex numerical and multi-step reasoning tasks by encouraging explicit intermediate reasoning steps [59]. These approaches have been further integrated into Retrieval-Augmented Generation (RAG) pipelines, where external evidence is retrieved and combined with structured reasoning to address numerical question answering and fact-checking tasks in domain-specific environments that exceed the model’s parametric knowledge [17]. In addition, test-time scaling techniques have been proposed to give a model more time to think, solving nuanced challenges by aggregating over multiple reasoning paths at inference time [38].

Despite these advances, modern LLM and RAG systems continue to exhibit frequent reasoning failures, particularly in real-world numerically grounded tasks [22, 46, 50]. In RAG-based numerical reasoning, models must correctly associate quantities with their originating sources, conditions, and constraints, often in the presence of noisy or heterogeneous retrieved evidence, which places particular demands on precise reasoning. Empirical studies show that even state-of-the-art models produce incorrect numerical conclusions, inconsistent intermediate steps, or answers that are unsupported by retrieved evidence [5, 2], as illustrated in Figure 1.1. These failures persist even when models generate long and seemingly coherent reasoning traces, suggesting that explicit reasoning alone does not guarantee correctness [22, 3].

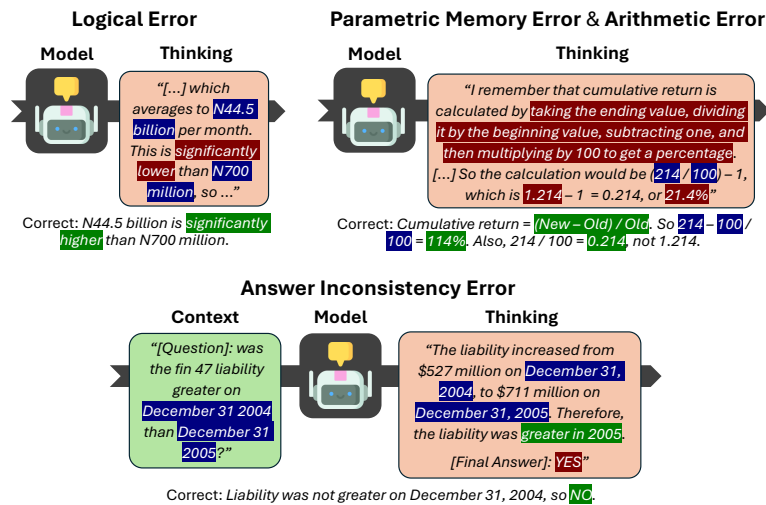


Figure 1.1: Examples of reasoning failures in LLM reasoning traces on open-domain numerical tasks, demonstrating the diversity of failure modes: (top left) a logical error where an incorrect magnitude comparison is drawn despite correct value retrieval; (top right) a parametric memory error, where an incorrect formula is used, and an arithmetic error, where the ensuing calculation is performed incorrectly; (bottom) an answer inconsistency error where a conclusion contradicts correct intermediate reasoning.

While a growing body of work catalogs reasoning errors in LLMs, most analyses remain largely surface-level and task-specific, offering limited organized insights [48, 2]. Existing taxonomies focus on surface-level phenomena such as hallucination, arithmetic errors, or faithfulness violations, but lack an overarching or intuitive typology that connects failure modes to underlying reasoning mechanisms [22]. As a result, many failure analyses stop short of identifying actionable causes that can inform model or system-level improvements.

Identifying and isolating specific reasoning failures is nonetheless critical for advancing LLM reliability. Fine-grained error characterization has previously proven effective in improving model robustness,

evaluation practices, and mitigation strategies across NLP tasks [66]. In the context of RAG-based numerical reasoning, understanding which reasoning steps fail, and how, is a necessary prerequisite for designing targeted interventions.

This work focuses on a particular but underexplored reasoning failure: *numerical scoping errors*. Scoping errors arise when a model incorrectly binds quantities, entities, or attributes to the wrong referents [29], for example, assigning a statistic to an incorrect population, time frame, or measurement unit. Figure 1.2 illustrates two such cases of numerical scoping errors: in the fact-checking example, the model conflates the scope of a percentage, reading *16% of Democratic-sponsored bills are substantive* when the correct reading is *16% of House-passed bills are Democratic-sponsored*; in the QA example, the model selects the wrong year’s balance from a table, confusing the 2014 and 2013 figures. Similar phenomena have been noted implicitly in work on entity tracking, numerical grounding, and context misalignment, but have not been systematically studied as a distinct failure class in RAG-based numerical reasoning [29, 16].

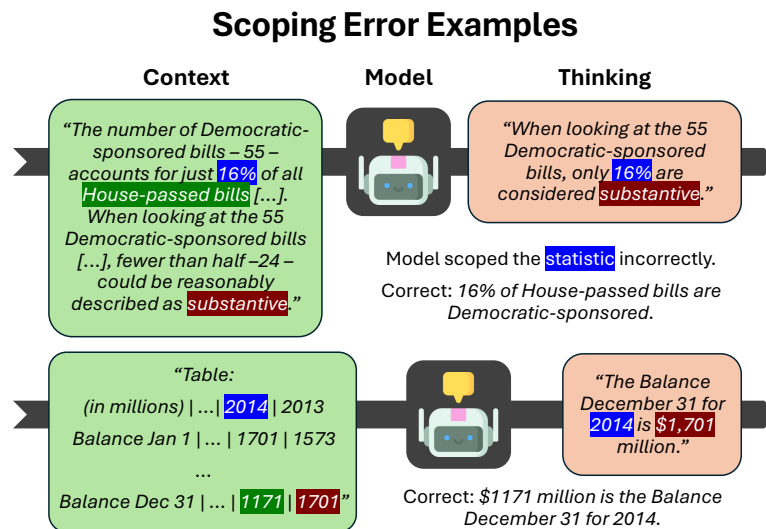


Figure 1.2: Numerical scoping error examples where a reasoning model (Deepseek-DistillQwen-7B; DeepSeek-AI et al. [13]) mis-scopes real-world statistics in the QUANTEMP numerical fact-checking dataset [53] and FinQA numerical question-answering (QA) dataset [9].

Correct contextual scoping is especially critical in RAG-based numerical reasoning tasks, including numerical question answering (QA) and numerical fact-checking (FC). In these settings, models must correctly associate numbers with their originating sources, conditions, and constraints, even among the natural noise retrieved. Errors in numerical scoping can lead to confidently stated but incorrect conclusions, as shown in Figure 1.2, even when all required information is present in the retrieved context. In high impact settings, a single mis-scoped number can change the meaning of a conclusion, turning a statistic into a misleading financial signal, an unsafe medical recommendation, or a false public claim, making the investigation of numerical scoping errors crucial for reliable LLM-based reasoning.

The goal of this research is to systematically characterize scoping errors in RAG-based numerical reasoning and to assess their role in broader reasoning failures. By treating scoping as a distinct and measurable reasoning concern, this work aims to bridge the gap between descriptive failure analyses and actionable improvements to numerical reasoning in LLMs and to demonstrate that failure characterization can serve as a productive methodology for addressing reasoning errors more broadly.

1.2. Research Questions

Specifically, this thesis aims to address the following research questions:

- RQ1. What is one of the most common reasoning failures in RAG-based numerical reasoning tasks?
- RQ2. How can scoping errors be reliably measured in RAG-based numerical reasoning settings?
- RQ3. How can scoping errors be mitigated to improve numerical reasoning performance in RAG systems?

1.3. Scientific Contributions

The contributions of this work, derived from addressing the proposed research questions, are as follows:

1. **Identified and characterized scoping errors in RAG-based numerical reasoning:** through qualitative & quantitative analysis of open-domain numerical reasoning traces, scoping errors are identified as a prevalent and pestilent reasoning failure. Through an in-depth literature study and further qualitative analysis, scoping errors are situated into a broad typology of reasoning failures and taxonomized into different types of scoping errors.
2. **Automatic structured measurement of scoping errors in RAG-based numerical reasoning:** numerical scoping errors are measured across open-domain numerical reasoning tasks using a hybrid LLM-as-a-Judge pipeline and more efficient distilled variants. Measurement methods are evaluated based on human agreement.
3. **Evaluation of scoping-error-aware mitigation strategies:** test-time scaling and reward modeling strategies are customized to target scoping errors. Scoping error rates are reduced, while modest downstream performance gains are observed. Potential limiting factors are identified and discussed.

The code used to implement the experiments is publicly available at <https://github.com/msmink01/numerical-scoping-errors.git> for reproducibility.

1.4. Thesis Outline

The report is laid out in the following way. Chapter 2 discusses current related work pertinent to the research objectives of this study. Chapter 3 lays out the methodology used to perform numerical reasoning inference, detect numerical scoping errors, and use scoping-specific signals to select optimal reasoning paths using test-time scaling techniques. Chapter 4 discusses the experiments conducted and the implementation specifications used to answer the research questions. Chapter 5 discusses the results obtained and situates them within the research questions and existing literature. Finally, Chapter 6 concludes the thesis while discussing limitations and directions for future improvements.

2

Related Work

2.1. Chain-of-Thought in LLMs

Chain-of-Thought (CoT) prompting was introduced as a simple but powerful mechanism to elicit intermediate reasoning steps in LLMs, improving performance on multi-step, arithmetic, symbolic, and logical reasoning [59]. Many works have adopted this approach and developed Large Reasoning Models (LRMs) trained or fine-tuned to explicitly reason before answering [46, 13]. Other works have extended CoT beyond text-only reasoning to multimodal settings [58] and to tool-augmented architectures such as retrieval in RAG pipelines [17, 2]. Despite these gains, CoT-based reasoning does not reliably guarantee correctness. Multiple studies demonstrate that detailed reasoning traces can coexist with incorrect conclusions, unsupported intermediate steps, and faithfulness violations [46, 22, 2]. In numerical reasoning tasks, models often produce coherent-looking chains of thought that nonetheless mishandle quantities, entities, or attributes [5]. This has led to growing skepticism that increased explicit reasoning monotonically improves accuracy, especially in real-world settings where context can be noisy, irrelevant, or conflicting [46, 2]. As a result, recent work increasingly treats CoT as a fallible reasoning process whose structure, length, and grounding require analysis.

2.2. Open-Domain Numerical Reasoning Tasks

Open-domain knowledge-intensive tasks require models to integrate external evidence with internal reasoning to complete objectives that exceed their parametric knowledge [17]. Numerical variants of these tasks introduce additional demands as models must correctly use quantities alongside their noisy originating sources, conditions, and constraints, as seen in Table 1.1. This form of open-domain mixed-modality reasoning differs from pure mathematical reasoning because the central challenge of the task is not only to compute correct results, but also to identify which retrieved quantities are relevant, preserve their source-specific context, and apply them under correct conditions.

FinQA [9] and TAT-QA [70] established numerical reasoning over hybrid tabular and textual financial documents as a benchmark challenge and have since been widely used to evaluate how LLMs handle multi-step arithmetic reasoning and evidence grounding across varying levels of complexity and reasoning depth [49, 5]. In the fact-checking setting, TabFact [8] established structured table-based claim verification, while QUANTEMP [53] and its extension [55] shift to open-domain numerical claim verification against noisy retrieved evidence, more closely reflecting real-world retrieval conditions. Recent work on QUANTEMP has explored evidence retrieval pipelines [39] and test-time scaling [11], demonstrating that both retrieval quality and reasoning strategy meaningfully affect numerical verification performance. Despite this, reasoning failures in these settings remain frequent and poorly characterized, motivating the focused analysis of this work.

2.3. Reasoning Failure Typologies in LLMs

Although most LRM works report LLM failure rates and discuss the limitations of LRM processing, few have proposed robust, generalizable, widely accepted reasoning failure typologies. Some works focus on specific reasoning failures. Hallucination, generated content that is nonsensical or unfaithful to the provided source content, has been widely recognized and studied in detail [22, 3]. Numerical scoping errors are related to hallucinations because both involve a mismatch in evidence-grounded reasoning. However, scoping errors are more specific: rather than fabricating a number, the model may copy a correct quantity but bind it to the wrong context or use it in an ungrounded way. In this sense, scoping errors can be locally grounded but globally unfaithful while hallucinations are often more global.

Other than hallucinations, there are many other investigated types of reasoning failures in open-domain numerical reasoning, but few are examined in the context of a broader failure typology. Arithmetic-specific and RAG-specific failure cases were studied in Boye and Moell [5] and Agrawal et al. [2], respectively. Despite focusing on different settings, both works show that failures often stem from incorrect problem representation, missed constraints, or plausible but unsupported reasoning paths. They differ mainly in emphasis: Boye and Moell [5] highlights failures in mathematical problem solving, while Agrawal et al. [2] highlights failures in question answering in RAG systems. These studies identify important recurring patterns, but present them as task-specific error categories with little organization rather than contextualizing them in a general reasoning failure typology.

Meanwhile, Song, Han, and Goodman [48] propose a broad reasoning failure typology that separates informal (intuitive) failures from formal (logical) failures. It further separates unembodied reasoning from embodied reasoning when interaction with a physical environment is (un)necessary. Although the informal failure typology is well developed, the formal failure typology lists separate logical failures with no real organization, placing a highly specific failure such as the reversal curse, where LLMs fail to extrapolate the converse of certain logical relations, at the same level as holistic, non-specific failures such as compositional reasoning. This work aims to build off of the reasoning failure typology proposed in Song, Han, and Goodman [48] and the individual failure characterization studies, introducing more organization and intuition while investigating the specific failure case explored in this work.

2.3.1. Scoping Errors

Scoping errors, or the incorrect binding of quantities, entities, or attributes to the wrong referents, have been indirectly studied and measured throughout LRM literature. Lepori, Mozer, and Ghandeharioun [29] frame contextualization errors as internal race conditions in the LLM where subject entities must be contextualized early in the forward inference process for correct inference. Feng and Steinhardt [16] analyze contextualization in language models (LMs), using causal mediation analysis to argue that LMs assign binding vectors to entity-attribute pairs, thus contextualizing the entity. Kamoi et al. [23] find that state-of-the-art LLMs struggle to detect context-faithfulness errors, a relative of scoping errors. Other works explore scoping errors indirectly by analyzing context-memory knowledge conflicts [62] or context inconsistency within faithfulness hallucination [22]. Most works directly examining scoping errors do so in synthetic settings with white-box models and rarely analyze *numerical* scoping. This work aims to specifically analyze and characterize numerical scoping errors in real-world numerical reasoning contexts.

2.4. Reasoning Failure Measurement Strategies in LLMs

Reasoning failure measurement in LLMs spans several levels of evaluation. At the output level, common metrics assess final answer correctness, completeness, faithfulness or groundedness to the provided evidence, and sometimes efficiency or conciseness of the reasoning process. More diagnostic analyses go beyond outcome quality and attempt to identify specific error types, such as hallucinations, unsupported claims, arithmetic mistakes, or other reasoning failures within generated traces.

Most works that identify and measure such specific reasoning failures in LLM outputs rely on human annotation, either through expert labeling of error types in reasoning traces [23], span-level localization of erroneous answer content [44], or manual qualitative analysis of reasoning traces [5, 2]. While reliable, human annotation is costly and difficult to scale, and error detection itself remains challenging even in benchmark settings with expert-labeled data [23]. Recent work has begun to automate error

classification in RAG settings [30], but tends to target system-level RAG error categories rather than fine-grained reasoning failures within individual traces.

The LLM-as-a-Judge paradigm [19] offers a scalable alternative by prompting an LLM to evaluate reasoning traces against defined criteria, and has been used to detect generated reasoning path errors in literature [11, 44, 30, 45]. However, Kamoi et al. [23] find low recall, unreliable explanations, and sensitivity to prompt changes across LLM-based error detectors, suggesting that general-purpose LLM judges remain unreliable for error detection. Structured rubric-based evaluation could improve detection reliability by decomposing evaluation into explicit sub-criteria assessed independently, as in fine-grained alignment evaluation frameworks [66]. In domain-specific settings, this approach has been applied to legal reasoning [28], demonstrating that structured criteria enable detection of failure types that holistic judges miss. This work applies a hybrid rubric-driven LLM-as-a-Judge pipeline to measure specifically numerical scoping errors, using human evaluation for ground truth calibration and automatic rubric-based measurement for scalable failure prevalence estimation.

2.5. Mitigation of Reasoning Failures in LLMs

Mitigation strategies for reasoning failures in LLMs can be grouped according to where they intervene in the generation pipeline. Retrieval-side approaches improve the quality, relevance, or structure of the evidence supplied to the model, which is central in RAG systems [6]. However, improving retrieval does not directly address failures in how the model reasons over retrieved evidence, which is the focus of this work. Reasoning-side failure mitigation approaches instead attempt to reduce errors by modifying the generation model, inputs, inference procedure, or outputs after generation.

Model-level approaches modify the model itself through supervised fine-tuning, preference optimization, or reward modeling. Preference-based methods such as Direct Preference Optimization (DPO) train models to prefer higher-quality outputs without explicitly fitting a separate reward model [42], while reinforcement learning methods use feedback from reward models to improve model behavior [4, 68]. Rubric-based reward modeling further attempts to make the feedback signal more explicit by decomposing broad response quality into criteria that can be optimized during training [61, 20]. These approaches usually optimize general helpfulness, correctness, or grounding rather than targeting a specific reasoning failure type and are computationally expensive to use.

Input-level approaches modify the problem formulation given to the model before generation. These methods include prompting strategies, problem decomposition, evidence restructuring, context compression, or explicit marking of important entities [55, 26, 7]. Prompting strategies can also improve reasoning by guiding the structure of intermediate steps [69, 10].

Inference-time approaches keep the generation model parameters and inputs fixed but alter how the model reasons at test time. Some methods introduce explicit iterative verification, self-correction, or retrieval loops [52, 14, 34], although other studies find that LLMs often fail to correct their own reasoning errors without stronger verification signals [24]. Test-time scaling is a related inference-time strategy that uses additional computation during inference, either within a single reasoning trajectory or across multiple candidate traces.

2.5.1. Test-Time Scaling and Reward-Guided Selection

Test-time scaling increases the amount of computation used at inference time rather than during training [38]. It can be broadly divided into sequential and parallel approaches [47]. Sequential test-time scaling allocates additional computation within a single reasoning trajectory, for example through iterative refinement or verification loops [47, 65, 14, 34]. Parallel test-time scaling instead samples multiple independent candidate reasoning traces and selects or aggregates among them after generation [47, 38]. Parallel approaches are especially common in the literature because they are simple to implement, naturally parallelizable, and directly compatible with answer aggregation methods such as self-consistency [57] and verifier-guided selection [32, 25, 67, 47, 61].

Self-consistency is a simple form of parallel test-time scaling in which multiple reasoning paths are sampled and their final answers are aggregated by majority vote [57]. Verifier-guided test-time scaling extends this idea by scoring or filtering candidate traces before selecting a final answer, as done by Chungkham et al. [11] in numerical fact-checking.

Verifier-guided selection often relies on reward models, which can be divided into Outcome Reward Models (ORMs) and Process Reward Models (PRMs) [32]. ORMs score a complete candidate output, usually according to final correctness or overall response quality. PRMs instead score intermediate reasoning steps, making them more fine-grained but also more expensive to train because they require step-wise process labels [32]. Recent work therefore explores methods for reducing the annotation cost of PRMs, including automatically generated or sample-efficient process supervision [25, 67, 47]. In contrast, ORMs are less fine-grained but easier to train and apply in parallel test-time scaling because they only require output-level labels.

This thesis follows the parallel test-time scaling setting. However, instead of using only generic correctness scores, this work uses scoping-error-aware signals to guide candidate trace selection in open-domain numerical reasoning. This positions the work as a targeted form of verifier-guided test-time scaling: the goal is not only to select traces that appear likely to be correct, but to prefer traces that avoid a specific measured reasoning failure, namely numerical scoping errors.

2.6. Position of This Work

The previous sections show that existing work has studied reasoning failures, numerical reasoning, open-domain reasoning, automatic error detection, and failure mitigation, but usually not in combination. Table 2.1 positions this thesis relative to representative prior work along dimensions central to the research questions, including task setting, type of failure analysis, and presence of detection or mitigation work. The comparison shows that existing studies usually cover only parts of the problem: they either analyze broad reasoning failures, study numerical tasks, classify open-domain errors, automatically measure failures, or mitigate failures at test-time. This thesis addresses this gap, operating at the intersection of these directions by investigating numerical scoping errors in open-domain numerical reasoning traces, detecting them automatically, and using scoping-aware signals for failure mitigation.

Table 2.1: Comparison of this thesis with representative related work across task setting, failure analysis, automatic detection, and mitigation dimensions.

Work	Task Setting		Failure Analysis		Automatic Failure Detection	Failure Mitigation
	Open-Domain	Numerical	Broad Typology	Specific Analysis		
Song, Han, and Goodman [48]: Reasoning failure survey	X	X	✓	X	X	✓
Huang et al. [22]: Hallucination survey	✓	X	✓	✓	✓	✓
Boye and Moell [5]: Mathematical reasoning failures	X	✓	X	✓	X	X
Lepori, Mozer, and Ghandeharioun [29]: Contextualization errors	X	X	X	✓	✓	✓
Kamoi et al. [23]: Error detection in LLM responses	✓	✓	X	✓	✓	X
Leung et al. [30]: RAG error classification	✓	X	✓	✓	✓	✓
Chungkham et al. [11]: Numerical claim verification TTS	✓	✓	X	✓	✓	✓
This work: Investigating open-domain numerical scoping errors	✓	✓	✓	✓	✓	✓

✓: central focus of the work; ✓: partially addressed, mentioned, or indirectly relevant; X: not a main focus.

3

Methodology

This chapter describes the methodology used to identify, measure, and mitigate numerical scoping errors in LLM reasoning traces. Its structure follows the research questions of this work. First, Section 3.1 introduces the inference pipelines used to generate reasoning traces for numerical veracity prediction and question answering. Section 3.2 then defines and characterizes numerical scoping errors within a broader reasoning failure typology. Section 3.3 describes how these errors are measured using human qualitative analysis, a straight LLM-as-a-Judge, and a hybrid LLM-as-a-Judge pipeline. Finally, Section 3.4 presents how the resulting scoping-error signals are used for mitigation through test-time scaling and learned reward models.

3.1. Inference Pipelines

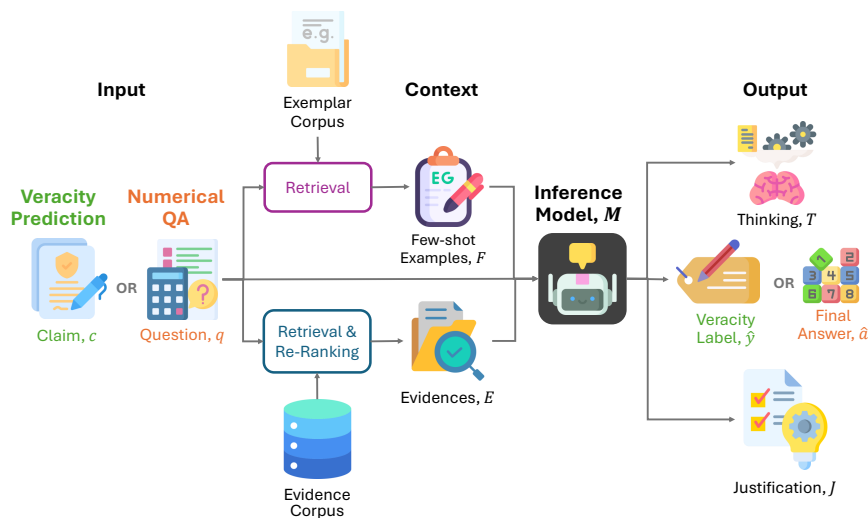


Figure 3.1: Overview of the inference pipelines used for veracity prediction (green) and numerical question answering (orange). In both settings, task inputs, few-shot examples, and evidence are provided to an inference model, which generates an intermediate reasoning trace, final prediction, and justification.

3.1.1. Veracity Prediction

The veracity prediction pipeline used for fact-checking numerical claims in the QUANTEMP dataset is presented in Figure 3.1. Given a claim c , the top- k evidence E is retrieved and re-ranked from an evidence corpus and the top- k few-shot examples F are retrieved from an exemplar corpus. Then the claim, few-shot examples, and evidence are fed to an inference model M , to generate the intermediate thinking trace T , veracity label \hat{y} , and final justification J , where $\hat{y} \in \{\text{True}, \text{False}, \text{Conflicting}\}$. The inference model is a pre-trained language model that exhibits language understanding, generation,

and reasoning capabilities. Formally, the fact-checking pipeline can be represented as:

$$(T, \hat{y}, J) = M(c, F, E) \quad (3.1)$$

3.1.2. Numerical Question Answering

The numerical question answering (QA) pipeline used for answering numerical questions from the FinQA dataset is presented in Figure 3.1. Given a question q , the relevant document pages E are provided from the source document, and the top- k few-shot examples F are retrieved from the exemplar corpus. The question, examples, and evidence are then fed to an inference model M , which generates an intermediate reasoning trace T , predicted answer \hat{a} , and final justification J . In contrast to veracity prediction, the output space of the final answer is not restricted to a fixed set of class labels. Instead, \hat{a} can be any numerical value or binary answer. The inference model is a pre-trained language model that uses the provided evidence to perform the required numerical reasoning and produce a final answer. Formally, the numerical QA pipeline can be represented as:

$$(T, \hat{a}, J) = M(q, F, E) \quad (3.2)$$

3.2. Reasoning Failure Typology for Numerical Reasoning in LLMs

This work uses a broad reasoning failure typology, refined from other works [48, 46, 22, 5, 2, 15], to situate numerical scoping errors within broader failures of LRM reasoning. Inspired by Song, Han, and Goodman [48], this typology distinguishes between intuitive reasoning failures and logical reasoning failures. This distinction is inspired by dual-process accounts of human reasoning [15] and provides a useful way to separate failures that appear to arise from heuristic, belief-driven behavior from failures that arise during explicit step-wise reasoning. For an overview of the typology and how scoping errors are situated, see Figure 3.2.

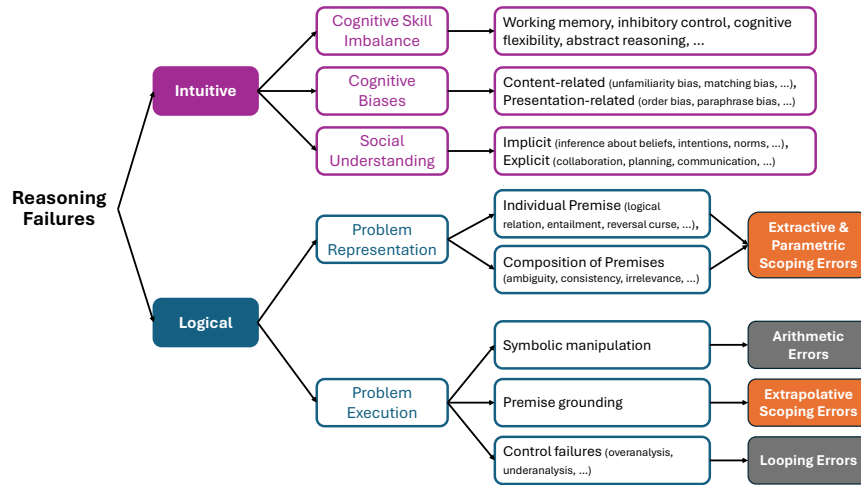


Figure 3.2: Overview of the reasoning failure typology, refined from [48, 46, 22, 5, 2, 15], used to contextualize numerical scoping errors. Failures are organized into intuitive and logical branches, with scoping errors positioned primarily as logical failures that can arise during problem representation or problem execution. Arithmetic and looping errors are shown as related but distinct failure modes.

3.2.1. Intuitive & Logical Reasoning Failures

Intuitive reasoning failures refer to cases where the model appears to rely on, and potentially misuse, salient cues, prior assumptions, familiar patterns, or presentation-dependent heuristics rather than task-specific processes, often in an impulsive or ungrounded manner. In numerical RAG settings, this may include cases where the model favors a familiar statistic, follows a misleading surface form, or relies on parametric knowledge despite contradictory retrieved evidence [5, 15, 48].

Logical reasoning failures refer to failures in the model’s explicit reasoning process. This work divides logical reasoning failures into two categories. First, problem representation failures occur when the

model constructs an incorrect or incomplete representation of the task or situation. These include failures in relevance identification, ambiguity resolution, consistency handling, logical relation understanding, and scoping [2, 22]. Second, problem execution failures occur when the model acts incorrectly on a represented problem, for example through arithmetic mistakes, unsupported premise use, constraint violations, or mis-allocated reasoning effort [46, 22].

3.2.2. Numerical Scoping Errors as Representation Failures

Numerical scoping errors are positioned primarily as problem representation failures. They occur when a model binds a numerical quantity, entity, attribute, unit, time frame, or condition to an incorrect referent. This means that the model may reason over a numerical value that is locally valid but globally misapplied to the wrong context. For instance, a quantity may be taken from the evidence correctly, but then attributed to the wrong year, company, country, subgroup, or claim component. This distinguishes scoping errors from other reasoning failures like arithmetic errors, where the relevant quantities are correctly scoped but the result is incorrectly computed.

The typology also captures why scoping errors can be difficult to isolate. A scoping error may originate from intuitive bias, such as over-reliance on a salient number, or from logical failure, such as losing track of a condition while combining multiple evidence statements. Once introduced, the error can propagate into later reasoning steps and produce final answers or veracity labels that appear justified by coherent but mis-scoped reasoning. For this reason, the typology treats scoping errors as potentially upstream failures that can affect subsequent numerical reasoning.

Types of Numerical Statistics in Model Reasoning

To analyze scoping errors more precisely, this work first distinguishes between three types of numerical statistics that may appear in model reasoning traces: extractive, extrapolative and parametric statistics.

Extractive statistics are numerical quantities that are taken directly from the task inputs, such as the claim, question, evidence, or source document. These statistics may be copied verbatim or paraphrased, but their numerical value is explicitly grounded in the provided input.

Extrapolative statistics are numerical quantities or outcomes derived from other statistics in the reasoning trace. These may result from arithmetic operations, comparisons, aggregations, unit conversions, or other reasoning steps that combine or transform extracted quantities.

Parametric statistics are numerical quantities introduced from the model's parametric memory rather than from the provided inputs or from a derivation over other quantities. These statistics are not directly supported by the claim, question, evidence, or source document, and are therefore treated as ungrounded unless they correspond to commonsense or task-independent knowledge required for interpretation.

Types of Numerical Scoping Errors

Based on these statistic types, this work distinguishes between three types of numerical scoping errors: extractive, extrapolative, and parametric scoping errors. See examples of each of these types of scoping errors in Table 3.1.

Extractive scoping errors occur when an input-grounded statistic is misinterpreted or misapplied in the model output. In these cases, the numerical value may be copied correctly, but its scope is incorrectly changed. For example, a statistic referring to one year, country, company, subgroup, condition, or measurement unit may be used as if it referred to another. These errors are especially important in RAG-based numerical reasoning because they can make the model appear evidence-grounded while still misrepresenting the evidence.

Extrapolative scoping errors occur when the model derives a numerical statistic or conclusion by improperly combining, manipulating, or comparing other statistics. In these cases, the source quantities may be individually grounded, but the operation connecting them is not valid under their respective scopes. For example, a model may compare values from different time periods, aggregate quantities from incompatible populations, or compute a difference between statistics that refer to different entities or units. The resulting derived statistic is therefore incorrectly scoped even if the individual input numbers are correct.

Parametric scoping errors occur when the model introduces a non-commonsense numerical statistic from parametric memory and treats it as factual within the task context, despite the statistic not being present in or derivable from the provided evidence. In these cases, the error is not simply that the model hallucinated a number, but that it assigned that number a task-specific scope, such as a particular unit, period, or population, without grounding. Parametric scoping errors are especially problematic in RAG settings because they can override or contaminate evidence-based reasoning.

Table 3.1: Examples of numerical scoping error types identified in model reasoning traces for numerical fact-checking and question-answering tasks. Mis-scoped numerical quantities are highlighted in red.

Scoping Error Type	Claim / Question	Evidence	Model Thinking	Explanation
Extractive: due to ambiguous or missing information in input	"Parliament is getting KSh38 billion, the judiciary is getting KSh18 billion, basically 50% [of parliament's allocation]."	"with regard to the judicial governance of the judiciary, the superior billion, with matters settled having a value of KSh554 million. "	"The evidence mentions the judiciary's allocation as KSh554 million "	The statistic is present in the evidence, but it is not clearly scoped to the judiciary's total allocation. The model therefore maps an ambiguous evidence quantity into a claim-relevant quantity.
Extractive: due to misunderstanding the input	"Says under his utility rate plan, 'An estimated 50 percent of our residential households will see a decrease in their water and wastewater bills.'"	–	"There's no data supporting a 50% decrease in bills."	The claim concerns the share of households expected to see a decrease, not the magnitude of the decrease. The model therefore mis-scopes the percentage from a population share to a reduction amount.
Extractive: due to ignoring existing information in the input	"Every taxpayer owes about \$130,000 to pay off the national debt."	"every american man, woman and child with a \$50,000 share of this \$16 trillion national debt"	"evidence states that every American is considered to owe a share of the \$16 trillion debt, but it doesn't specify the exact amount per person. "	The evidence does specify a \$50,000 per-person share. The model therefore fails to preserve the scope of the amount as an individual share.
Extrapolative: due to combining incompatible time scopes	"Nearly 1,000 people were shot and killed by police in the U.S. between April 2020 and April 2021. "	"seventy-three U.S. police officers were gunned down in 2021 . During that same year, officers shot and killed roughly sixteen times more civilians." " 1,021 people were shot and killed by police in 2020. "	The model derives approximately 1,168 civilians killed in 2021 from 73×16 , then adds this to the full-year 2020 figure: "Adding those together gives us about 2,189 people for the period. "	This mis-scopes full-year statistics to the partial period between April 2020 and April 2021 , leading to an overestimate of the true value.
Extrapolative: due to comparing incompatible time frames	"In 1999 Lagos was generating N700 million in internally generated revenue per month."	"lagos generated N267 billion in the first six months of 2021 , a monthly average of N44.5 billion. "	"Lagos generated N267 billion in the first six months of 2021 , which averages to N44.5 billion per month. This is significantly lower than N700 million. "	Aside from the arithmetic issue of naming 44.5 billion lower than N700 million, the reasoning mis-scopes a 2021 revenue statistic as evidence for a 1999 revenue claim.
Parametric: due to an ungrounded parametric statistic introduced as factual	"Molenbeek city in Belgium has become the first European city with 100 per cent Muslim population. By 2021 Belgium will become an Islamic country."	"percent of the brussels population and 7.5 percent of the antwerp population is muslim"	"The evidence say that in Brussels and Antwerp, only 2.5% and 7.5% of the population are Muslim."	The 2.5% statistic is not present in the evidence and is not derived from another input statistic. The model therefore introduces an ungrounded parametric statistic and scopes it to Brussels.
Parametric: due to a parametric statistic being used without considering temporal scope	"There are 278 Republicans in Congress. With Eric Cantor's defeat, they are now all Christian and all white except for one black senator, who was appointed."	–	"I know the U.S. Congress has 254 Republicans and 281 Democrats. So, 278 Republicans is incorrect."	The comparison relies on a parametric statistic that is not grounded in the provided evidence and is used without regard to the relevant time frame of the claim.

3.2.3. Relevance of Typology & Scoping Error Characterization

In the remainder of this work, the reasoning failure typology serves as the conceptual basis for identifying and analyzing numerical scoping errors in reasoning traces generated for veracity prediction and numerical QA. Furthermore, the distinction of the different kinds of scoping errors informs future processes for scoping error detection and analysis.

3.3. Measuring Numerical Scoping Errors

Numerical scoping failures are difficult to measure because they are not directly observable from the final answer alone: a reasoning trace can contain a scoping error while still producing the correct final prediction. This necessitates the inspection of the claim/question, the provided evidence, and the intermediate reasoning trace jointly. Furthermore, scoping errors often co-exist, are exacerbated by, and exacerbate other reasoning failures such as arithmetic errors, reasoning control errors, and consistency errors. The process of detecting scoping errors is cognitively demanding for human annotators and difficult to accomplish with automatic judgement.

This work therefore combines human qualitative analysis with LLM-as-a-Judge-based measurements. Human analysis is used to ground the definition of scoping errors, estimate their prevalence in sampled traces, and motivate the distinction between extractive, extrapolative, and parametric scoping errors. LLM-as-a-Judge methods are then used to scale the measurement beyond the manually inspected examples. Two judge-based scoping detection approaches are considered: a straight judge baseline and a hybrid judge pipeline. The straight judge provides a simple baseline, while the hybrid pipeline decomposes scoping error detection into traceable statistic-level decisions.

3.3.1. Human Qualitative Analysis

Human qualitative analysis was used to identify and characterize reasoning failures in generated model traces. Samples were selected using stratified random sampling. For QUANTEMP, samples were stratified by claim label and claim taxonomy class. For FinQA, samples were stratified by answer type and value. This ensured that the analysis covered different task formats, answer forms, and numerical reasoning settings.

The annotation target was the generated reasoning trace rather than only the final prediction. Annotators inspected the task input, evidence, reasoning trace, final answer or label, and justification. To identify the prevalence of some reasoning failures, annotators marked whether the trace contained a numerical scoping error, arithmetic error, or looping error. A scoping error was identified when a numerical quantity was bound to an incorrect referent during reasoning. This made it possible to distinguish scoping errors from arithmetic errors, where quantities are incorrectly manipulated, and looping errors, where the model impulsively repeats or cycles through reasoning without progress. See Appendix A for the guidelines human annotators used to detect each type of error.

The qualitative analysis served three purposes. First, it provided an estimate of scoping error prevalence in model reasoning traces. Second, it informed the typology of extractive, extrapolative, and parametric scoping errors. Third, it provided qualitative grounding for the design and validation of the automatic measurement pipeline.

3.3.2. Straight LLM-as-a-Judge

As a simple automatic baseline, this work uses a straight LLM-as-a-Judge approach. Given the task input claim or question x , evidence E , reasoning trace R , and ground truth answer y , a judge model J directly predicts whether the trace contains numerical scoping errors \hat{s}_R , arithmetic errors \hat{a}_R , and looping errors \hat{l}_R :

$$(\hat{s}_R, \hat{a}_R, \hat{l}_R) = J(x, E, R, y) \quad (3.3)$$

where $\hat{s}_R, \hat{a}_R, \hat{l}_R \in \{0, 1\}$ indicates whether the reasoning trace is judged to contain at least one scoping, arithmetic, or looping error, respectively.

This approach is simple and scalable, but coarse. For finding scoping errors, a direct judge must identify relevant quantities, infer their intended scoping, compare their use in the reasoning trace against the input, and decide whether an error occurred in a single step. As a result, the judgement is difficult to interpret and does not explicitly identify which statistic caused the error. Furthermore, the direct judge is directly given the ground truth answer, and thus is unusable for unlabeled inputs. For these reasons, the direct judge is treated as a baseline rather than the primary measurement method.

3.3.3. Hybrid LLM-as-a-Judge Scoping Failure Detection Pipeline

The primary scoping error detection method is a multi-step hybrid LLM-as-a-Judge pipeline. The design is based on the observation that numerical scoping errors are usually tied to specific statistics in the reasoning trace. Instead of judging the full trace in one step, the pipeline identifies statistics, classifies their source, verifies their scoping, and aggregates statistic-level judgements into a trace-level scoping error label.

Given a task input claim or question x , evidence E , and reasoning trace R , the pipeline first identifies a set of candidate statistics Q_R :

$$Q_R = \{q_1, q_2, \dots, q_n\}. \quad (3.4)$$

Each statistic q_i is then assigned a type τ_i :

$$\tau_i \in \{\text{extractive, extrapolative, parametric}\}. \quad (3.5)$$

Finally, each statistic is evaluated for whether it is used with the correct scope:

$$\hat{s}_i = J_{\tau_i}(q_i, x, E, R), \quad (3.6)$$

where $\hat{s}_i \in \{0, 1, \text{unknown}\}$ denotes whether statistic q_i is judged to be scope-valid, mis-scoped, or unknown, respectively. The type-specific judge J_{τ_i} applies different criteria depending on whether the statistic is extractive, extrapolative, or parametric. Figure 3.3 presents an overview of this pipeline, Figure 3.4 contains an example of the pipeline processing a numerical QA query, and more details are provided in each subsection below.

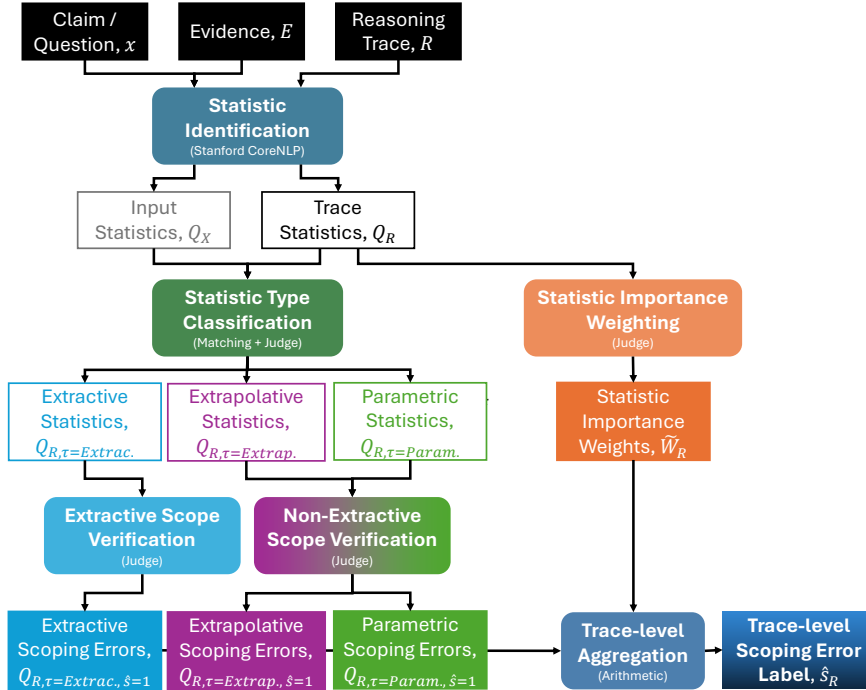


Figure 3.3: Overview of the hybrid LLM-as-a-Judge pipeline for measuring numerical scoping errors. The pipeline extracts statistics from the task inputs and reasoning trace, classifies trace statistics as extractive, extrapolative, or parametric, verifies their scope using type-specific judge calls, and combines statistic-level scoping judgements with normalized importance weights to produce a trace-level scoping error label.

Statistic Identification

Statistics are identified from both the output reasoning trace and the task inputs including the claim/question and evidence. The reasoning trace statistics are the main objects checked for scoping errors, while the input statistics are used to determine whether a reasoning trace statistic is extractive.

Numerical quantities are first extracted using the Stanford CoreNLP `parse` annotator [36]. The extracted quantities are then grouped and de-duplicated using information from the Stanford CoreNLP `tokenize`, `ssplit`, `pos`, `lemma`, and `ner` annotators [36]. This produces a dictionary of candidate statistics, where each group contains different mentions of the same or highly similar statistic. The output of this step is a set of reasoning trace statistics Q_R and a corresponding set of input statistics Q_X .

Statistic Type Classification

Each reasoning trace statistic is classified according to its source. First, the statistic is compared with the input statistics using the linguistic and similarity features extracted during statistic identification. If the statistic matches an input statistic, it is labeled as extractive and enters the extractive verification branch. If no match is found, the statistic enters the non-extractive branch.

In the non-extractive branch, an LLM-as-a-Judge determines whether the statistic is extrapolative or parametric. A statistic is labeled as extrapolative if it is derived or computed from other statistics, for example through arithmetic, comparison, aggregation, or transformation. A statistic is labeled as parametric if it is introduced as an assumption or from the model's parametric memory rather than from the provided input or a derivation of other statistics.

Scope Verification

Scope verification is performed using type-specific LLM-as-a-Judge calls. For extractive statistics, the judge receives the statistic, reasoning trace, and task inputs. It is instructed to locate the exact input sentence or sentences containing the statistic, then evaluate whether the statistic is used in the reasoning trace with the same meaning and scope. The statistic is marked as mis-scoped if its unit, scale, referent, time frame, population, condition, or surrounding context differs from the input. It is marked as scope-valid only when the input fully supports the statistic's use in the reasoning trace without added assumptions.

For non-extractive statistics, the judge first confirms whether the statistic is extrapolative or parametric. If the statistic is extrapolative, the judge identifies the input quantities used to derive it and checks whether the derivation is valid under their scopes. An extrapolative statistic is marked as scope-valid if the derivation is mathematically valid, the input quantities are compatible, and the resulting statistic is representative of its inputs without unjustified unit, assumption, or scope mixing. It is marked as mis-scoped if incompatible units are combined, if the derivation is ungrounded, or if the derived quantity is applied outside the scope supported by its inputs.

If the statistic is parametric, the judge evaluates whether the statistic is acceptable background knowledge or an unsupported factual claim. Parametric statistics are marked as scope-valid when they are commonsense, generic, or explicitly stated as assumptions rather than task-specific facts. They are marked as mis-scoped when they are specific, hard-to-know factual numbers introduced without evidence, especially when they are assigned to a particular scope.

In all verification branches, the judge may return an unknown label when the statistic cannot be confidently evaluated. The full prompts used for each judge call are provided in Appendix A.

Trace-level Aggregation with Statistic Weighting

After scope verification, statistic-level judgements are aggregated into a trace-level scoping error label. A simple aggregation rule would mark a trace as containing a scoping error if any statistic is mis-scoped. However, this can overestimate trace-level scoping failures when a mis-scoped statistic is incidental, redundant, or unimportant to the reasoning. To reduce this issue, the pipeline first assigns an importance weight to each statistic.

An LLM-as-a-Judge receives the list of statistics, reasoning trace, and task inputs, and assigns an importance weight w_i to each statistic. The judge is instructed to give high weight to statistics whose removal or modification would likely change the final answer or justification, and low weight to statistics

that are merely background, redundant, or briefly mentioned. Since the raw judge-assigned weights w_i may not sum to exactly one, they are automatically normalized as:

$$\tilde{w}_i = \frac{w_i}{\sum_{j=1}^n w_j}, \quad (3.7)$$

where n denotes across all the statistics for that reasoning trace. This ensures that the normalized weights \tilde{w}_i sum to one:

$$\sum_{i=1}^n \tilde{w}_i = 1. \quad (3.8)$$

A trace R is then marked as containing a scoping error if at least one sufficiently important statistic is judged to be mis-scoped:

$$\hat{s}_R = \mathbb{I}(\exists i : \tilde{w}_i \geq \lambda \wedge \hat{s}_i = 1), \quad (3.9)$$

where \tilde{w}_i is the normalized importance weight for statistic q_i , λ is the importance threshold, and $\hat{s}_i = 1$ indicates that the statistic q_i is mis-scoped. This produces an interpretable statistic-level explanation and a trace-level scoping error label \hat{s}_R . The importance weighting prompt is provided in Appendix A.

Figure 3.4 depicts an example of the pipeline used on a numerical QA query.

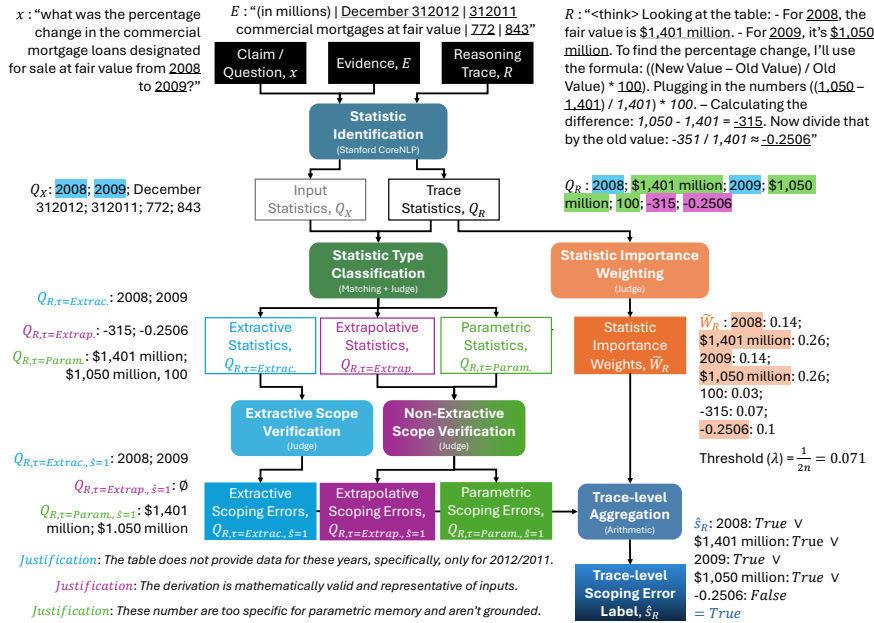


Figure 3.4: Example of scoping detection by hybrid LLM-as-a-Judge pipeline with a numerical QA query from FinQA [9]. The pipeline extracts statistics from the task inputs x , E and reasoning trace R , classifies trace statistics Q_R as extractive, extrapolative, or parametric, verifies their scope using type-specific judge calls, and combines statistic-level scoping judgements with normalized importance weights to produce a *True* trace-level scoping error label.

3.3.4. Distilling the Hybrid LLM-as-a-Judge Pipeline into a Student Model

The hybrid LLM-as-a-Judge pipeline involves multiple judge calls and computationally intensive statistic-level tasks. To make the measurement of scoping errors more efficient, this work explores the distillation of scoping-error labels from the hybrid LLM-as-a-Judge pipeline into smaller student models, further detailed in Section 3.4.3.

3.4. Mitigating Scoping Errors with Parallel Test-Time Scaling

The final part of the methodology evaluates whether numerical scoping errors can be mitigated through parallel test-time scaling. Instead of changing the inference model parameters, parallel test-time scaling generates multiple candidate reasoning traces for the same task instance and then selects or aggregates among them [38]. This allows the system to use external verification signals to prefer traces that are more likely to be correct or less likely to contain scoping errors.

3.4.1. Candidate Trace Generation

Given a fact-checking claim or question-answering question x , few-shot examples F , and evidence E , the inference model M generates K candidate outputs:

$$\mathcal{O}_K = \{o_1, o_2, \dots, o_K\}, \quad (3.10)$$

where each output o_k consists of an intermediate thinking trace T_k , final prediction \hat{p}_k , and justification J_k . For veracity prediction, the final prediction is a label $\hat{p}_k \in \{\text{False}, \text{True}, \text{Conflicting}\}$. For numerical QA, the final prediction is an answer \hat{p}_k , which may be numerical or binary. The mitigation methods differ in how they select a final output from \mathcal{O}_K .

3.4.2. Scoping Error Filtering with the Hybrid LLM-as-a-Judge Pipeline

In order to integrate scoping-aware scores, scoping-aware mitigation methods use the hybrid LLM-as-a-Judge pipeline H_{scope} from Section 3.3.3 as a reward model. For each candidate trace o_k , the scoping verifier produces a trace-level scoping-error label:

$$\hat{s}_k = H_{scope}(x, E, o_k), \quad (3.11)$$

where $\hat{s}_k = 1$ indicates that candidate trace o_k is judged to contain a sufficiently important scoping error, and $\hat{s}_k = 0$ indicates that no such scoping error is detected. Candidate outputs are filtered by removing traces with detected scoping errors:

$$\mathcal{O}_K^{scope} = \{o_k \in \mathcal{O}_K : \hat{s}_k = 0\}. \quad (3.12)$$

This filtering method directly implements the hypothesis that removing traces with detected scoping errors should improve downstream prediction quality. This method can then be combined with other test-time scaling methods such as self-consistency or correctness verifiers to select the best reasoning traces without scoping errors.

3.4.3. Learned Scoping-Aware Reward Models

The hybrid judge pipeline provides an interpretable process-level signal but requires multiple judge calls and statistic-level analysis. To test whether this signal can be approximated more efficiently, this work also trains outcome reward models (ORMs) from scoping detection pipeline-derived labels.

Scoping ORM

The scoping ORM is trained to predict whether a candidate output contains a scoping error. Unlike the hybrid scoping detection pipeline, which evaluates individual statistics and aggregates their labels, the scoping ORM S_ϕ directly scores the full output:

$$z_k = S_\phi(x, T_k, \hat{p}_k, J_k), \quad (3.13)$$

where z_k is the predicted scoping-error logit for candidate output o_k . Given the binary scoping target $\hat{s}_k \in \{0, 1\}$, where $\hat{s}_k = 1$ indicates that the hybrid judge pipeline detected a scoping error, the scoping ORM is trained using binary cross entropy (BCE) with logits:

$$\mathcal{L}_{\text{scope}} = -\frac{1}{N} \sum_{k=1}^N [\hat{s}_k \log \sigma(z_k) + (1 - \hat{s}_k) \log(1 - \sigma(z_k))], \quad (3.14)$$

where N is the number of candidate outputs in the batch and $\sigma(\cdot)$ is the sigmoid function. An important note is that this model is not provided the evidence as one of its inputs, limiting its ability to detect extractive scoping errors, but lowering the context length demands and computational costs for the model. An ablation of this no-evidence design choice is presented in Appendix B. At inference time, the scoping ORM can be used either to filter traces predicted to contain scoping errors or to rank traces by predicted scoping quality. This tests whether a distilled reward model can capture the original hybrid pipeline’s scoping signal without running the full hybrid pipeline. The prompt for this ORM can be seen in Appendix A and further training details are provided in Section 4.5.

Multi-Task Correctness & Scoping ORM

Finally, this work evaluates a multi-task ORM trained to predict both general output quality and scoping error risk. The motivation is that selecting traces solely for low scoping error risk may not be sufficient if other failure modes, such as arithmetic errors or unsupported reasoning, remain. A multi-task reward model V_ψ therefore produces both a correctness logit r_k and a scoping logit z_k :

$$(r_k, z_k) = V_\psi(x, T_k, \hat{p}_k, J_k). \quad (3.15)$$

The multi-task ORM is trained with two BCE losses: a correctness loss $\mathcal{L}_{\text{correctness}}$ and a scoping-error loss $\mathcal{L}_{\text{scope}}$. Given the binary correctness target $c_k \in \{0, 1\}$ and scoping-error target $\hat{s}_k \in \{0, 1\}$, the two losses are computed separately for the correctness and scoping heads. The final multi-task objective combines them by unweighted addition, $\mathcal{L}_{\text{multi}} = \mathcal{L}_{\text{correctness}} + \mathcal{L}_{\text{scope}}$. An ablation for this multi-task objective can be seen in Appendix B.

During inference, the multi-task model can select traces using a joint score:

$$\hat{p}_{\text{multi}} = \arg \max_{\hat{p}_k: o_k \in \mathcal{O}_K} (\sigma(r_k) - \alpha \sigma(z_k)), \quad (3.16)$$

where \hat{p}_{multi} denotes the prediction from the selected candidate trace given the combined scores and α controls the penalty assigned to predicted scoping errors. The combined model can also be used to select traces using approximated scoping filtering and correctness Best-of-N:

$$\hat{p}_{\text{multi}} = \arg \max_{\hat{p}_k: o_k \in \tilde{\mathcal{O}}_K^{\text{scope}}} \sigma(r_k), \quad (3.17)$$

where $\tilde{\mathcal{O}}_K^{\text{scope}}$ represents the set of candidate traces without a scoping error according to the multi-task ORM:

$$\tilde{\mathcal{O}}_K^{\text{scope}} = \{o_k \in \mathcal{O}_K : \sigma(z_k) \leq \lambda\}. \quad (3.18)$$

for some scoping threshold λ . Like the single-task scoping ORM, the multi-task ORM is not given the evidence as input, limiting its ability to recognize extractive scoping errors, but lowering its context length demands and computational cost. The multi-task ORM mitigation variant tests whether general correctness verification is more useful when combined with a scoping-aware signal, even when implemented efficiently. The remaining training details for this model can be seen in Section 4.5.

3.4.4. Overview of Scoping-Aware Mitigation Variants

Table 4.2 and Figure 4.1 contain overviews of all of the evaluated mitigation variants, including those that use scoping-aware signals. Overall, scoping-aware signals are emitted by the hybrid LLM-as-a-Judge scoping detection pipeline from Section 3.3.3 and distilled either in a single-task ORM or multi-task ORM, allowing the detection and handling of scoping errors in reasoning traces.

4

Implementation Details & Experiments

4.1. Experimental Overview

This chapter describes the implementation details and experimental setups used to instantiate the methodology introduced in Chapter 3, including dataset configurations, model settings, training setups, evaluation metrics, hardware configurations, and more.

The experiments are organized around two main goals. First, measurement experiments estimate the prevalence of reasoning failures, with a particular focus on numerical scoping errors, in model-generated reasoning traces. These experiments include human annotation, a straight LLM-as-a-Judge baseline, and the hybrid scoping error measurement pipeline, all described in Section 3.3. Second, mitigation experiments evaluate whether test-time scaling and scoping-aware reward signals, detailed in Section 3.4, can improve final task performance or preserve correct candidate traces.

4.2. Datasets & Evidence Configuration

Since the goal of this work was to evaluate numerical reasoning failures in real-world settings, datasets were selected that require numerical reasoning across different task formats and real-world evidence paradigms. QUANTEMP was used for numerical fact-checking, while FinQA was used for numerical question answering. The two tasks differ in their output format, evidence structure, and reasoning requirements, allowing scoping errors to be studied across both classification-style and open-answer outputs, tabular and textual evidence, and verification- and reading-comprehension-style numerical reasoning.

4.2.1. QUANTEMP

The QUANTEMP dataset [53] is a numerical fact-checking dataset comprised of 15,514 numerical claims sourced from various fact-checking platforms such as Politifact, Snopes, AFP, and others. Each numerical claim is associated with gold standard evidence from human fact-checkers and categorized into one of four claim taxonomies: statistical, temporal, interval, and comparison claims. The dataset also includes an evidence corpus of 423,320 snippets. QUANTEMP++ extends this setting by refining the evidence corpus, retrieving open-domain evidence using decomposed claims, and reducing temporal and gold-evidence leakage [55].

In this work, four evidence configurations are used for QUANTEMP-based experiments. The *Gold* setting refers to the human fact-checker explanations provided with the original dataset. The *Re-Ranked* setting refers to the top-3 evidence snippets from the QUANTEMP evidence corpus after top-100 BM25 and top-3 re-ranking with the `all-MiniLM-L12-v2` cross-encoder model [56]. The *Filtered Search Results* setting refers to the top-10 cleaned open-domain BM25 evidence snippets from QUANTEMP++. Finally, the *Qrels* setting refers to the final open-domain evidence recommended by QUANTEMP++ [55]. Both QUANTEMP++ evidence configurations are included because the more processed *Qrels* setting does not

always contain evidence for every claim. Due to computational and human-resource limits, only the *Gold* & *Re-Ranked* evidence settings are used for human evaluation and test-time scaling experiments.

The QUANTEMP dataset is split into train, validation, and test sets containing 9,935, 3,084, and 2,495 claims, respectively. The label distribution consists of 18.79% true claims, 57.93% false claims, and 23.27% conflicting claims. The claim taxonomy distribution consists of 7,302 statistical claims, 4,193 temporal claims, 2,357 interval claims, and 1,645 comparison claims.

For the main evaluation experiments, performance is reported on the QUANTEMP test split. Few-shot examples are retrieved from the QUANTEMP validation split using top-2 BM25. For reward model training, additional generated traces are created from sampled training queries as described in Section 4.5.1. Human qualitative analysis used a stratified sample of generated traces, as described in Section 4.4.1.

4.2.2. FinQA

The FinQA dataset [9] is a numerical question answering dataset focused on financial reasoning over real-world financial reports. It consists of 8,281 question-answer pairs derived from publicly available financial documents. Each question requires multi-step numerical reasoning over semi-structured evidence such as tables and accompanying textual descriptions, and often involves arithmetic operations, aggregations, or comparisons.

In this work, three evidence configurations are used for FinQA. The *Whole-Document* setting includes the full page-level evidence associated with the question. The *Re-Ranked* setting uses the retrieved evidence subset provided by the dataset, and the *Gold* setting uses the human-annotated supporting evidence [9]. Due to computational and human-resource limits, only the *Whole-Document* evidence settings are used for human evaluation and test-time scaling experiments.

FinQA is split into 6,251 training set instances, 883 validation set instances, and 1,147 test set instances. The usage of these splits in the experiments is identical to QUANTEMP. In contrast to QUANTEMP, FinQA does not define a fixed set of class labels. Instead, answers consist of numerical values or binary yes/no responses. For this reason, FinQA predictions are evaluated using exact match after answer normalization, as described in Section 4.7.1.

4.3. Model & Generation Setup

4.3.1. Inference Model

Because the analysis in this work requires access to intermediate reasoning traces, all inference experiments use generative language models rather than classification-only natural language inference models. Classification models commonly used for fact-checking, such as RoBERTA-Large-MNLI or BART-MNLI, return final labels but do not expose the intermediate reasoning process needed for scoping error analysis. Therefore, the inference setup is restricted to language models that generate both a reasoning trace and a final prediction.

The main inference model used in this work is Deepseek-Distill-Qwen-7B with $Q6_K$ quantization [13]. This model was selected because of its accessible size, relatively widespread prevalence in LRM literature, and ability to produce explicit reasoning traces suitable for process-level analysis. The model is used to generate reasoning traces, final labels or answers, and final justifications for both veracity prediction and numerical QA. To assess whether the observed results depend on this specific generation model, most experiments were additionally repeated with Deepseek-Distill-Llama-8B with $Q6_K$ quantization; ablation results for this secondary inference model can be found in Appendix B.

Inference was performed through LMStudio using a local API [51]. Unless otherwise specified, the inference model was run with a context length of 8,192 tokens, a maximum generation length of 2,048 tokens, top-p of 0.9, temperature of 0.2, frequency penalty of 0, repeat penalty of 1.0, and flash attention enabled. For test-time scaling experiments, stochasticity was increased by sampling candidate traces using temperatures uniformly distributed across 0.2, 0.3, 0.4, and 0.5.

4.3.2. Judge Model

The LLM-as-a-Judge experiments use Qwen-2.5-14B-Instruct as the LLM judge [63]. This model is used for both the straight LLM-as-a-Judge baseline and the hybrid judge pipeline. An instruction-

tuned model was selected for judge tasks because these tasks require following structured prompts and returning parseable outputs. The judge model is also distinct from the main inference models, which reduces the risk of self-confirmation bias caused by using the same or very similar model family for both generation and evaluation.

The judge model was run through LMStudio with $Q4_K_M$ quantization, a context length of 16,384 tokens, and a maximum generation length of 2,048 tokens. Since judge outputs should be deterministic and schema-compliant, the temperature was set to 0.0. Other decoding parameters were kept consistent with the inference model where applicable. Invalid judge outputs were retried with a maximum generation length of 4,096 tokens.

4.3.3. Reward Models

Learned outcome reward models (ORMs) were initialized from Llama-3.2-3B [18] and hosted using HuggingFace’s `transformers` library [60]. Three reward model variants were trained: a correctness ORM, a scoping ORM, and a multi-task correctness and scoping ORM. The correctness ORM predicts general output quality or correctness to rank candidate traces, the scoping ORM predicts whether a trace contains a scoping error, and the multi-task ORM jointly predicts correctness and scoping error risk scores.

For the single-task ORM variants, a simple linear prediction head was added on top of the base model to return one logit. For the multi-task ORM, two separate linear prediction heads were added on top of the shared base model, with one head predicting correctness and the other predicting scoping error risk. The reward model inputs were truncated to a maximum sequence length of 1,024 tokens. This truncation occasionally removed later parts of the thinking part of long reasoning traces, but most traces were within 1,024 tokens.

4.3.4. Prompt Templates & Output Formats

All prompt templates are provided in Appendix A. The prompts were designed to produce structured outputs that could be parsed automatically. For inference, the prompts instructed the model to generate an intermediate reasoning trace followed by a final prediction and justification, separated by standardized tags. For fact-checking the final prediction was one of `True`, `False`, or `Conflicting`. For numerical QA, the final prediction was a numerical value or a binary yes/no answer. Judge prompts were designed to return structured JSON outputs so judge decisions could be extracted programmatically.

When an output could not be parsed into the required schema, it was marked as invalid. Invalid outputs were excluded from metrics requiring a valid prediction and were reported separately through invalidity rates. This distinction prevents malformed outputs from being silently ignored while still allowing performance metrics to be computed over valid predictions.

4.4. Scoping Error Measurement Implementation

4.4.1. Human Annotation Setup

Human annotation was used to qualitatively validate the reasoning failure categories and provide a reference point for evaluating automatic judge-based measurement. A total of 79 generated reasoning traces were annotated. The sample consisted of 41 `FinQA` traces with *Whole-Document* evidence and 38 `QUANTEMP` traces with *Re-Ranked* evidence, selected using stratified sampling to cover both task types and a range of answer categories.

One annotator labeled all 79 traces for the presence of three reasoning failures: numerical scoping errors, arithmetic errors, and looping errors, to assess the relative prevalence of each error type in numerical reasoning. See Appendix A for the guidelines used to identify each error type. The scoping error labels were further used to evaluate the performance of the automatic scoping measurement methods.

To assess agreement on scoping error labels, the scoping error annotation task was also assigned to four additional annotators. After completing a small trial annotation round to assess understanding, it was the idea that each additional annotator would label approximately half of the 79 traces for the presence or absence of scoping errors. However, due to non-responses, only one additional annotator

completed their labeling. Because there was only one additional annotator, agreement was computed between the primary annotator and the secondary annotator on the half of the queries that had been double labeled. Cohen’s kappa [12] was used to quantify pairwise agreement for the binary scoping error labels, and was 0.312. This agreement is not very high, but upon manual inspection of labels, disagreements were resolved and labels were updated.

4.4.2. Straight LLM-as-a-Judge Setup

The straight LLM-as-a-Judge setup was implemented to explore the relative prevalence of various reasoning failures for further analysis and as a coarse automatic baseline for reasoning failure detection. The judge model received the task claim or question, evidence, generated reasoning traces, final prediction, final justification, and ground-truth answer. It was then asked to determine whether the trace contained a numerical scoping error, arithmetic error, or looping error, and whether each detected error directly contributed to an incorrect answer.

The straight judge prompt did not include few-shot examples. This choice was made to avoid making the judge overly sensitive to the specific failure patterns shown in examples and artificially inflating or deflating estimated error prevalence. The full prompt template is provided in Appendix A.

The output of the straight judge was parsed into structured fields indicating whether each failure type was present and whether it affected the final answer. Outputs that could not be parsed into the required structure were marked as invalid. Across the entire test sets, the average invalidity rates of `QUANTEMP` and `FinQA` across all evidence splits were 0.01% & 0.15%, respectively. The straight judge was evaluated through human verification of a stratified subset of 79 judged traces, where judge decisions were compared against human labels for the presence and type of reasoning failures.

4.4.3. Hybrid Judge Pipeline Implementation

The hybrid LLM-as-a-Judge pipeline implements the scoping error measurement method introduced in Section 3.3.3. The pipeline was designed to produce more interpretable scoping error labels by decomposing trace-level judgement into statistic-level extraction, classification, verification, and aggregation. The same judge model as in the straight judge baseline, `Qwen-2.5-14B-Instruct` [63], was used for the hybrid pipeline.

Statistic Extraction & Grouping

Candidate statistics were extracted from both the reasoning trace and task inputs. The reasoning trace statistics are the quantities evaluated for scoping correctness, while the input statistics are used to determine whether a trace statistic is extractive. Trace statistics were only extracted from the intermediate thinking reasoning trace, not the final label or justification, to avoid duplicate statistics and to focus the scoping error detection on the reasoning process itself. Input statistics were extracted from the claim or question and evidence, while few-shot examples were excluded to avoid matching statistics from unrelated problem settings.

Statistic extraction was implemented using a local Stanford CoreNLP server with the `tokenize`, `ssplit`, `pos`, `lemma`, `ner`, and `parse` annotators [36]. A timeout of 600,000 milliseconds was used for CoreNLP statistic identification to enable the processing of long traces and evidence. Before parsing, text was normalized to improve quantity extraction. This normalization removed model-specific reasoning tags, standardized quotation marks, inserted spacing between numbers and adjacent unit, currency, or percentage symbols, and collapsed whitespace. For `FinQA`, table rows were deterministically converted into natural-language sentences before extraction so that quantities in structured tables could be processed by the same text-based extraction procedure.

Quantities were identified from CoreNLP constituency parses [36]. Candidate quantity mentions were anchored on numerical cardinal number nodes, labeled `CD` by CoreNLP’s constituency parser, and expanded to the nearest relevant quantity or noun phrase, labeled `QP` or `NP`. This allowed the extractor to retain local context around numbers, for example distinguishing a bare number from a phrase such as “50 percent”, “29 seasons”, or “N44.5 billion per month”. When the reasoning trace or evidence contained a high density of numbers and mathematical symbols, the text was split into smaller chunks using punctuation before parsing to reduce CoreNLP parsing failures.

After extraction, candidate numerical quantity mentions were normalized and grouped. Each mention was converted into a normalized representation containing its quantity type, numerical value, unit, approximation marker, and remaining unit-like text. Quantity types included percentages, monetary values, general numbers, and other number-like expressions. Mentions were grouped when they had compatible quantity types and sufficiently similar normalized values. Approximate quantities, such as those introduced by words like "about", "roughly", or "nearly", were matched using a looser tolerance than exact quantities. This produced a de-duplicated dictionary of input statistics Q_X and reasoning trace statistics Q_R , where each statistic group preserved its quantity text, sentence location, sentence context, and normalized metadata.

Statistics Type Classification & Scope Verification

Each reasoning trace was first compared against the extracted input statistics. If a sufficiently similar input statistic was found, the reasoning statistic was treated as extractive and pushed to the extractive verification branch. If no input match was found, the statistic was passed to the non-extractive branch, where the judge determined whether it was extrapolative or parametric and verified its scope. Each statistic-level verification returned a scope-valid, mis-scoped, or unknown label. To avoid overloading the judges with too many statistics, reasoning trace statistics were passed to the judges in a batch size of 20 and the results were aggregated. Exact prompts can be found in Appendix A.

Trace-Level Aggregation

After statistic-level verification, the pipeline aggregated statistic-level labels into a trace-level scoping error label. To avoid marking an entire trace as erroneous due to an incidental or unimportant statistic, each statistic was assigned an importance weight by a judge. The predicted weights were automatically normalized to sum to one. A trace was marked as containing a scoping error if at least one mis-scoped statistic exceeded the importance threshold. The threshold was set dynamically per trace as:

$$\lambda = \frac{1}{2n}, \quad (4.1)$$

where n is the number of unique reasoning trace statistics being compared. This arbitrary threshold marks a statistic as important if its normalized importance weight is at least half of the uniform weight that would be assigned if all statistics were equally important. This threshold was arbitrarily chosen after observing the relative prevalence of human-found and pipeline-found scoping failures. If any intermediate step of the pipeline returned unparseable results, that step was first retried with a larger maximum output length. Then, if the intermediate step still returned unparseable results, those statistics were excluded from trace-level label aggregation.

The trace-level label produced by the hybrid judge pipeline was used in two ways. First, it served as a measurement of scoping error prevalence in reasoning traces. Second, it was used as a supervision signal for the learned scoping reward models described in Section 4.6.3. The hybrid judge pipeline was evaluated through human verification of a stratified subset of 79 judged traces, where judge decisions were compared against human labels for the presence of scoping failures.

4.4.4. Distilled Scoping Measurement

Using the scoping error labels from the hybrid LLM-as-a-Judge pipeline, several student models were taught to recognize scoping errors in a more efficient manner. Model architecture and training procedures are detailed in Sections 4.3.3 and 4.5, respectively. During inference, a reasoning trace is marked with a scoping error if the student model's score is above an arbitrary threshold λ of 0.4. This threshold was chosen arbitrarily after observing the relative prevalence of teacher-found and student-found scoping failures. See Appendix B for ablations of λ . This signal is compared to the hybrid judge pipeline and used for downstream performance estimation.

4.5. Reward Model Training

4.5.1. Training Data Construction

Reward models were trained using generated reasoning traces on training set instances with varying temperatures. This allowed reward models to learn from variation across multiple candidates for the

same underlying query. Training data was constructed separately for QUANTEMP & FinQA with the *Re-Ranked & Whole-Document* evidence splits, respectively. Sample reward model prompt templates are provided in Appendix A.

For each dataset, 2,000 training queries were sampled using stratified sampling. These queries were split into 1,700 training queries and 300 mini-validation queries. Stratification was based on answer type and answer for FinQA and taxonomy label and final label for QUANTEMP to preserve representative diversity in the sampled training data. For each query, 12 candidate traces were generated using Deepseek-Distill-Qwen-7B. Candidate traces were sampled with temperatures uniformly distributed across 0.2, 0.3, 0.4, and 0.5.

After generation, traces were de-duplicated to remove near-identical outputs. De-duplication was performed using cosine similarity (Appendix A) between embeddings produced by the all-MiniLM-L12-v2 embedding model [56]. Traces with cosine similarity greater than 0.97 were treated as duplicates. This process removed around 29% of FinQA traces, and 20% of QUANTEMP traces.

The remaining traces were then labeled for correctness. Correctness labels were determined by matching each candidate final prediction against the ground truth label using the same validation procedure used for outcome-level evaluation, detailed in Section 4.7.1. The final datasets were sampled from the remaining traces based on correctness label with stratified sampling, with the aim of having a training dataset of around 10,000 traces.

For FinQA the final reward-model training and mini-validation sets contained 10,829 and 1,902 traces, respectively. For QUANTEMP, the final training and mini-validation sets contained 10,924 and 1,934 traces, respectively. For an overview of the correctness and scoping distributions per set, see Table 4.1.

Table 4.1: Distribution of reward model training data after trace generation, de-duplication, and stratified sampling based on correctness labels.

Dataset	Split	Total traces	Correct	Incorrect	Scope-Valid	Mis-Scoped
FinQA	Train	10,829	4,846	5,983	8,317	2,512
FinQA	Mini-validation	1,902	722	1,180	1,458	444
QUANTEMP	Train	10,924	3,658	7,266	7,285	3,639
QUANTEMP	Mini-validation	1,934	640	1,294	1,350	584

4.5.2. Training Procedure

All reward models were trained using parameter-efficient fine-tuning with LoRA [21]. The LoRA rank was set to 8 and LoRA alpha was set to 16. Models were trained with a batch size of 12 using an AdamW optimizer [33], a learning rate of 1×10^{-4} , epsilon of 1×10^{-8} , and random seed of 42. Training used the PyTorch package implementation of binary cross entropy (BCE) loss for logits [40]. The maximum number of epochs was set to 10, with early stopping patience of 2 based on loss on the mini-validation set. A cosine learning rate scheduler was used with a warm-up period equal to 5% of the total number of training steps.

To combine the correctness-ranking and scoping-error objectives in the multi-task ORM, this work performed ablations between straight task combination and Soft Optimal Uncertainty Weighting (UW-SO) [27] in Appendix B. For each batch, the separate BCE losses $\mathcal{L}_{\text{correctness}}$, $\mathcal{L}_{\text{scope}}$ were computed for the correctness and scoping heads. In straight task combination, the final loss $\mathcal{L}_{\text{multi}}$ was:

$$\mathcal{L}_{\text{multi}} = \mathcal{L}_{\text{correctness}} + \mathcal{L}_{\text{scope}}.$$

In UW-SO, losses were detached and converted into inverse loss scores so the lower-loss task received larger softmax weights:

$$\mathbf{w} = \text{softmax} \left(\frac{1}{\tau} \left[\frac{1}{\text{stopgrad}(\mathcal{L}_{\text{correctness}}) + \epsilon}, \frac{1}{\text{stopgrad}(\mathcal{L}_{\text{scope}}) + \epsilon} \right] \right), \quad (4.2)$$

where $\mathbf{w} = [w_{\text{correctness}}, w_{\text{scope}}]$, $\tau = 5.0$ is a softmax temperature to allow more even weighting, and $\epsilon = 1 \times 10^{-8}$ avoids division by zero errors. The final training objective in UW-SO was the weighted

sum of the two original non-detached losses:

$$\mathcal{L}_{\text{multi-uwso}} = w_{\text{correctness}} \mathcal{L}_{\text{correctness}} + w_{\text{scope}} \mathcal{L}_{\text{scope}}.$$

The weights $w_{\text{correctness}}$, w_{scope} were re-computed for each batch. Single task ORMs used only the BCE loss for training, as defined in Section 3.4.3.

For the correctness ORM, the binary target indicated whether the candidate trace produced the correct final answer or label. For the scoping ORM, the binary target indicated whether the candidate trace contained a detected scoping error according to the hybrid judge pipeline. For the multi-task ORM, the model jointly optimized correctness prediction and scoping error prediction using two prediction heads. The training process recorded average training loss, training accuracy, validation loss, and validation accuracy for each epoch. Learning curves of each ORM during training can be seen in Appendix B.

4.6. Parallel Test-Time Scaling Experiments

To evaluate scoping-aware verification signals, downstream performance of candidate trace selection was compared with various verification signal baselines.

4.6.1. Test-Time Candidate Trace Generation

At test time, multiple candidate reasoning traces $\mathcal{O}_K = \{o_1, o_2, \dots, o_K\}$ were generated for each query, where each output o_k consists of an intermediate thinking trace T_k , final prediction \hat{p}_k , and justification J_k . For QUANTEMP, candidate sets were generated with up to $K = 12$ traces per claim with the *Gold* and *Re-Ranked* evidence splits, yielding 119,760 generated candidate reasoning traces across inference models. For FinQA, candidate sets were generated with up to $K = 20$ traces per question with the *Whole-Document* evidence split, yielding 45,880 generated candidate reasoning traces across inference models. The larger maximum K value was used for FinQA because the FinQA test set is smaller than the QUANTEMP test set, making the additional generation computationally feasible.

Candidate traces were sampled using temperatures uniformly distributed across 0.2, 0.3, 0.4, and 0.5. For QUANTEMP, results are reported for $K \in \{1, 4, 8, 12\}$. For FinQA, results are reported for $K \in \{1, 4, 8, 12, 16, 20\}$. For each value of K , aggregation methods operated only on the first K generated candidate traces for each instance. Outputs that could not be parsed into the required schema were marked as invalid. Invalid candidate traces were excluded from aggregation methods that required a valid final prediction and were reported separately through invalidity rates.

4.6.2. Baseline Scoping-Blind Aggregation Methods

Oracle Upper Bound

An oracle upper bound is used to estimate the maximum possible benefit of selecting among K traces. The oracle marks a task instance as correct if any valid candidate output contains the correct final prediction:

$$\text{Oracle}(\mathcal{O}_K) = \mathbb{I}(\exists o_k \in \mathcal{O}_K^{\text{valid}} : \hat{p}_k = p^*), \quad (4.3)$$

where $\mathcal{O}_K^{\text{valid}}$ denotes the subset of outputs with validly parsed predictions and p^* is the ground-truth label or answer. This is not a deployable method because it uses the gold answer, but acts as an upper bound by estimating whether the generated candidate set contained a correct answer that a better selector could have chosen.

Self-Consistency

Self-consistency is used as a primary test-time scaling baseline [57]. It selects the final prediction by majority vote over valid candidate outputs. The self-consistency prediction \hat{p}_{SC} is:

$$\hat{p}_{SC} = \text{mode}\{\hat{p}_k : o_k \in \mathcal{O}_K^{\text{valid}}\}. \quad (4.4)$$

Self-consistency tests whether sampling multiple traces improves performance through answer agreement alone, without using any verifier.

Correctness ORM Baseline

A generic outcome reward model (ORM) is used as a learned ranking baseline. Given task claim or question x , and candidate trace intermediate thinking T_k , predicted output \hat{p}_k , and justification J_k , the correctness ORM R_θ produces a correctness logit r_k which can be converted into a score through the sigmoid equation $\sigma(\cdot)$ estimating the general output quality or correctness of the candidate trace:

$$r_k = R_\theta(x, T_k, \hat{p}_k, J_k). \quad (4.5)$$

The selected output \hat{p}_{ORM} is the prediction of the highest-scoring candidate:

$$\hat{p}_{ORM} = \arg \max_{\hat{p}_k: o_k \in \mathcal{O}_K} \sigma(r_k). \quad (4.6)$$

This baseline tests whether a general correctness learned verifier can improve trace selection without explicitly targeting numerical scoping errors.

4.6.3. Scoping-Aware Test-Time Scaling Variants

In order to integrate scoping-aware scores, scoping-aware mitigation methods use the hybrid LLM-as-a-Judge pipeline from Section 3.3.3 or distilled variants as a reward model to remove or de-prioritize candidate outputs with sufficiently important scoping errors.

Oracle After Scoping Filtering

An oracle upper bound is computed after scoping filtering. This oracle measures whether the subset of traces judged to be free of scoping errors still contains a correct answer:

$$\text{Oracle}(\mathcal{O}_K^{\text{scope}}) = \mathbb{I}(\exists o_k \in \mathcal{O}_K^{\text{scope}} : \hat{p}_k = p^*), \quad (4.7)$$

where $\mathcal{O}_K^{\text{scope}}$ denotes the subset of outputs with validly parsed predictions without a sufficiently important scoping error. Comparing this oracle to the unfiltered oracle estimates whether the scoping filter removes correct candidates, preserves correct candidates, or leaves enough correct candidates for downstream selection.

Self-Consistency After Scoping Filtering

Self-consistency can also be applied after scoping filtering. Instead of voting over all valid traces, the final prediction is selected by majority vote over the subset of traces without detected scoping errors:

$$\hat{p}_{\text{scope}+SC} = \text{mode}\{\hat{p}_k : o_k \in \mathcal{O}_K^{\text{scope}}\}. \quad (4.8)$$

This method tests whether majority voting becomes more reliable after traces containing detected scoping errors are removed.

Ranking After Scoping Filtering

The generic correctness ORM can also be combined with scoping filtering. First, the scoping verifier removes traces with detected scoping errors. Then, the correctness ORM R_θ selects the highest-scoring remaining candidate:

$$\hat{p}_{\text{scope}+ORM} = \arg \max_{\hat{p}_k: o_k \in \mathcal{O}_K^{\text{scope}}} \sigma(R_\theta(x, T_k, \hat{p}_k, J_k)). \quad (4.9)$$

This method tests whether scoping filtering and generic output-quality ranking provide complementary selection signals.

4.6.4. Overview of Mitigation Variants

Table 4.2 contains an overview of the mitigation variants. Overall, the mitigation methods compare three kinds of selection signals. The first relies only on answer agreement, as in self-consistency. The second uses generic correctness rewards, as in the correctness ORM. The third uses scoping-error-aware signals, either through the hybrid scoping detection pipeline, a distilled scoping ORM, or a multi-task ORM; for an overview, see Figure 4.1. Oracle variants are included as upper bounds to distinguish between two possible limitations: whether correct answers are absent from the sampled traces, or whether they are present but not selected by the verifier or aggregation strategy.

Table 4.2: Overview of test-time scaling variants used to evaluate numerical scoping error mitigation.

Variant	Type	Selection Signal	Purpose
Oracle	Upper Bound	Gold answer availability among sampled traces	Estimates whether any sampled trace contains the correct answer.
Hybrid Pipeline: Scoping Filtering + Oracle	Scoping-Aware Upper Bound	Gold answer availability after scoping filtering	Estimates whether the scoping filter preserves correct candidate traces.
Self-Consistency	Baseline	Majority vote over valid answers	Tests whether answer agreement alone improves performance.
Hybrid Pipeline: Scoping Filtering + Self-Consistency	Scoping-Aware Baseline	Majority vote after scoping filtering	Tests whether voting improves after removing mis-scoped traces.
Correctness ORM: Correctness Best-of-N	Baseline	Learned general output-quality score	Tests whether generic reward-based trace selection improves performance.
Hybrid Pipeline: Scoping Filtering + Correctness ORM: Correctness Best-of-N	Scoping-Aware	Correctness ORM after scoping filtering	Tests whether generic reward-based trace selection improves after removing mis-scoped traces.
Scoping ORM: Scoping Filtering + Correctness ORM: Correctness Best-of-N	Scoping-Aware Single-Task Learned Verifier	Distilled scoping-error filtering before Correctness ORM	Tests whether hybrid scoping judgements can be approximated by a learned reward model.
Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	Scoping-Aware Multi-Task Learned Verifier	Joint distilled scoping-error filtering and correctness scores	Tests whether general quality and scoping-specific signals can be distilled into one model.
Multi-Task ORM: Correctness + Scoping Best-of-N	Scoping-Aware Multi-Task Learned Verifier	Joint correctness and scoping scores	Tests whether ranking based on scoping and correctness scores improves over filtering by scoping then ranking by correctness.

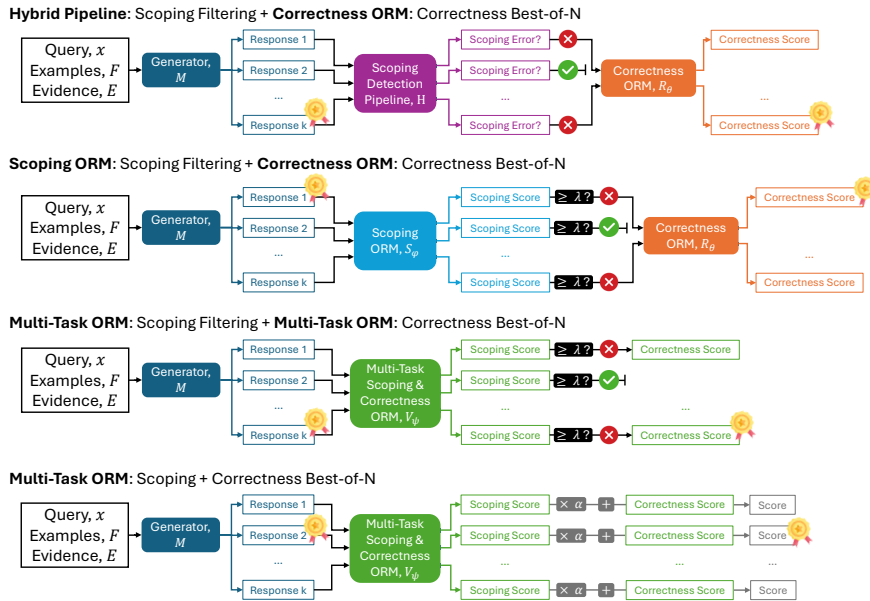


Figure 4.1: Overview of the scoping-aware test-time scaling variants used to evaluate numerical scoping error mitigation.

4.6.5. Test-Time Scaling Fallback Rules

The test-time scaling experiments compare several aggregation and selection variants. When a selection variant produced a tie, ties were resolved deterministically according to the order in which candidate traces were generated. This tie-breaking rule is arbitrary, but was applied consistently across all aggregation variants. If all generated traces for an instance were invalid or all traces were filtered out by scoping filtering, the instance was marked as invalid for that subset of deployable aggregation methods and not included in final result metrics, instead being reported in an invalidity rate.

4.7. Evaluation Metrics

Evaluation was performed at several levels to capture both final task performance and intermediate reasoning behavior. Outcome-level metrics evaluate whether the final model prediction is correct for each task. Reasoning-failure metrics measure the prevalence and pestilence of reasoning failures in reasoning traces. Agreement metrics evaluate how closely automatic judges and distilled student models reproduce human or teacher labels. Finally, invalidity, diversity, runtime, and statistical testing metrics are used to interpret test-time scaling behavior, efficiency, and the reliability of associations.

4.7.1. Downstream Outcome-Level Metrics

Outcome-level downstream performance metrics differed by task and followed the metrics used in the corresponding dataset papers [53, 9]. For numerical fact-checking on QUANTEMP, performance was measured using weighted-F1, macro-F1, and per-class F1 over verdict labels. Weighted-F1 accounts for class imbalance by weighting each class-specific F1 score according to its support, while macro-F1 gives equal weight to each class. In some analyses, these metrics were also reported separately across claim taxonomy categories. For more details on F1 scores see Appendix A.

For numerical QA on FinQA, downstream outcome-level performance was measured using exact-match (EM) accuracy over valid model responses. Since FinQA answers may be binary or numerical, predictions were evaluated according to answer type. For binary questions, the predicted answer was counted as correct only when it exactly matched the ground-truth answer when lower-cased. For numerical answers, predictions and labels were first normalized by removing common formatting symbols such as currency symbols, percentage signs, and commas. Following V., Prabhu, and Anand [54], a numerical prediction was counted as correct if it approximately matched the ground-truth value under either of two tolerance checks: the absolute rounded values matched within a relative tolerance of 0.02 at three decimal places, or the absolute values matched after rounding to one decimal place.

4.7.2. Reasoning Failure Prevalence & Pestilence

Reasoning failure prevalence was measured as the proportion of valid reasoning traces judged to contain a given error type. For human annotation and straight LLM-as-a-Judge analyses, prevalence was reported for numerical scoping, arithmetic, and looping errors across correct and incorrect traces. For the hybrid LLM-as-a-Judge pipeline, prevalence was reported at the trace-level, indicating whether a trace contained at least one sufficiently important detected scoping error, and at the statistic-level, indicating the average proportion of extracted quantities judged to be mis-scoped per statistic type.

Pestilence measured whether an identified reasoning failure contributed to an incorrect final answer. In the human annotation and straight LLM-Judge analyses, this was reported as the proportion of detected failures judged to contribute to final-answer incorrectness. This distinction was important because a reasoning trace can contain a scoping error without the final answer necessarily being incorrect.

4.7.3. Human Annotator, LLM Judge, & Distilled Student Agreement Metrics

Several agreement metrics were used to evaluate whether automatic judges and student models reproduced human or teacher labels. For trace-level scoping error detection, weighted-F1 was used to compare hybrid judge predicted binary scoping labels against human annotations while accounting for class imbalance. Furthermore, the average proportion of correctly identified, classified, and verified statistics per reasoning trace was also reported to show where the hybrid pipeline was making mistakes.

For student distillation performance, ROC-AUC was used to measure how well the student model ranked traces with and without scoping errors according to the teacher or human labels (Appendix A). ROC-AUC was included because the student ORMs produce continuous scoping scores, not only binary labels. Label agreement was then computed after thresholding these scores into binary scoping-error predictions. Label agreement was computed as the percentage of traces for which the predicted binary scoping label matched the reference label. Because label agreement can be inflated by majority-class predictions, especially when scoping error prevalence is imbalanced, majority-class baselines were also reported. Teacher scoping rate and student scoping rate were also reported to show whether student models reproduced the overall prevalence of scoping errors or behaved conservatively by predicting fewer scoping errors than the teacher hybrid judge or human annotators.

4.7.4. Invalidity, Diversity & Runtime Metrics

Invalidity rates were computed for generated inference outputs and judge outputs. For inference outputs, invalidity measured the proportion of candidate traces for which no valid final answer or label could be parsed. For judge outputs, invalidity measured the proportion of judge calls that did not return a parseable response. Reporting invalidity is important because both test-time scaling and judge-based measurement depend on valid structured outputs and the other metrics are unable to report invalid outputs without contaminating other results.

Candidate diversity was measured during test-time scaling. Since test-time scaling can only improve performance when the candidate set contains meaningful variation, answer-level diversity metrics were used to quantify variation across the K generated traces. Diversity was measured using the normalized answer entropy and answer disagreement (Appendix A). These metrics were used to interpret whether increasing K produced genuinely diverse candidate answers or merely repeated similar predictions.

When comparing the teacher and student models, runtime efficiency was measured using relative speed between the hybrid LLM-as-a-Judge pipeline and distilled student ORMs. Relative speed was computed as the average runtime of the hybrid pipeline divided by the average runtime of the student model over the same test dataset on the same hardware. This metric was used to quantify the practical speedup obtained by distilling the hybrid pipeline’s scoping signal into smaller reward models.

4.7.5. Statistical Testing

Statistical tests were used for selected comparisons to assess whether observed differences were likely to reflect systematic effects rather than sampling noise. Because test-time scaling produces multiple traces for the same query, trace-level observations are not independent. To account for this repeated-measures structure, generalized estimating equations (GEE) [71] were fit with query identifier as the clustering variable and an exchangeable working correlation structure. This models the average population-level association while allowing traces from the same query to be correlated.

For binary outcomes, such as whether a trace was correct, logistic GEE models were used. For example, to test whether scoping errors were associated with final correctness, the marginal mean model was:

$$\text{logit}(P(\text{correct}_{ij} = 1)) = \beta_0 + \beta_1 \text{scoping}_{ij}, \quad (4.10)$$

where correct_{ij} indicates whether trace j for query i was correct, and scoping_{ij} indicates whether the trace contained a scoping error. The coefficient β_1 was reported using its odds ratio, 95% confidence interval, and p-value.

For continuous outcomes, such as testing whether correctness ORM scores vary significantly with scoping error prevalence, Gaussian GEE models were used:

$$\text{score}_{ij} = \beta_0 + \beta_1 \text{scoping}_{ij} + \epsilon_{ij}. \quad (4.11)$$

Here, score_{ij} is the correctness ORM score for trace j of query i . In this setting, β_1 represents the estimated average difference in correctness ORM score associated with the presence of a scoping error. Results were reported with 95% confidence intervals and p-values, alongside descriptive group means.

4.8. Hardware & Software Configuration

Inference experiments were run on a local workstation equipped with an NVIDIA RTX 5090 GPU and an Intel(R) Core(TM) Ultra 9 285K processor. Local language model inference was served through LMStudio [51] using local API endpoints. Reward model training was performed on DelftBlue [1], a supercomputer cluster at TU Delft, using an NVIDIA A100 80GB GPU. The implementation was written in Python version 3.10.9 [41], with PyTorch [40] and HuggingFace’s transformers [60] libraries used for reward model training. Parameter-efficient fine-tuning was implemented using PEFT [35]. Stanford CoreNLP version 4.5.10 [36], which is Java-based, was run as an external local service for statistic extraction and parsing.

5

Results

5.1. RQ1: Identifying A Common Reasoning Failure in RAG-based Numerical Reasoning

5.1.1. Generating Reasoning Traces Across Tasks

In order to identify a common reasoning failure in RAG-based numerical reasoning tasks, reasoning traces were generated by `Deepseek-Distill-Qwen-7B` across `QUANTEMP` and `FinQA` evidence splits, as defined in Section 4.2. The downstream performance of these reasoning traces is presented in Tables 5.1 and 5.2.

From Table 5.1, fact-checking performance is strong with *Gold* evidence but drops in all open-domain evidence settings, falling below the fine-tuned NLI classifiers despite having generated reasoning traces. Across the open-domain evidence settings, the generative model is weakest on Interval claims, with particularly low scores for the *Qrels* and *Re-Ranked* evidence classes. By verdict class, the weakest scores are seen for the *True* class, while in the *Gold* setting, the *Conflicting* verdict class is the most difficult.

Inference Model	Evidence Category	Statistical		Temporal		Interval		Comparison		Per-Class			Total	
		M-F1	W-F1	M-F1	W-F1	M-F1	W-F1	M-F1	W-F1	T-F1	F-F1	C-F1	M-F1	W-F1
Naive (Majority class) [†]	-	22.46	34.25	28.35	62.95	25.86	49.19	16.51	16.32	0.00	72.64	0.00	24.21	41.42
Fine-tuned RoBERTa-Large-MNLI [†]	<i>Gold</i>	60.87	65.44	66.63	81.11	58.35	69.56	60.74	60.36	56.86	82.92	48.79	62.85	69.79
Zero-shot gpt-3.5-Turbo [†]	<i>Gold</i>	53.40	57.51	50.88	69.15	50.97	62.10	51.05	49.56	56.77	75.35	28.00	53.37	60.47
Fine-tuned RoBERTa-Large-MNLI [†]	<i>Qrels</i>	-	-	-	-	-	-	-	-	53.62	81.39	36.95	57.28	65.41
Fine-tuned RoBERTa-Large-MNLI [†]	<i>Re-Ranked</i>	49.55	52.48	60.29	74.29	48.84	57.93	40.72	39.66	51.59	70.60	35.27	52.48	58.52
Few-shot Deepseek-Distill-Qwen-7B	<i>Gold</i>	68.87	72.09	63.18	68.43	67.33	71.74	68.53	74.48	73.68	80.43	47.39	67.17	71.28
	<i>Qrels</i>	35.97	37.21	36.54	38.44	31.76	34.69	33.42	34.85	29.50	39.35	37.15	35.33	36.76
	Filtered search results	35.99	36.19	39.75	41.34	34.40	36.17	32.63	32.85	34.70	38.43	36.52	36.55	37.27
	<i>Re-Ranked</i>	39.32	39.80	34.78	36.26	32.78	34.93	36.91	39.51	35.28	40.27	35.40	36.98	38.16

[†] These results are reported directly from the `QUANTEMP(++)` publications [53, 55] and are included for reference only. They are not directly comparable to the other rows due to differences in model architecture, training data, and evaluation protocol.

Table 5.1: Fact-checking outcome-level performance across claim taxonomy categories, verdict categories, and evidence classes on the `QUANTEMP` test set. Occasionally, reasoning trace outcomes were invalid (0.4-2.2% of traces), resulting in exclusion.

For `Deepseek-Distill-Qwen-7B`, `FinQA` performance is comparatively stable across evidence settings, with *Re-Ranked* evidence slightly outperforming the other settings, which likely follows from the high degree of similarity between the evidence settings as each is highly likely to contain the gold information needed to answer the claim with fewer distractors when approaching the *Gold* evidence setting. Unlike `QUANTEMP`, the more distracting evidence settings do not appear to substantially hurt the generative model, although the higher invalidity rate makes these exact-match results less straightforward to interpret.

Inference Model	Evidence Category	Exact Match Accuracy
Fine-tuned RoBERTa-Large [†]	Gold	70.00
Fine-tuned RoBERTa-Large [†]	Re-Ranked	61.24
Few-shot Deepseek-Distill-Qwen-7B	Gold	66.94
	Re-Ranked	69.02
	Whole-Document	67.92

[†] These results are reported directly from the FinQA publication [9] and are included for reference only. They are not directly comparable to the other rows due to differences in model architecture, training data, and evaluation protocol.

Table 5.2: Question-answering outcome-level performance across evidence classes on the FinQA test set. Reasoning trace outcomes were sometimes invalid (22.7-27.3% of traces), resulting in exclusion.

In general, these results informed subsequent decisions for which evidence splits to focus human effort on when looking for common reasoning failures. Specifically, for QUANTEMP, the *Re-Ranked* evidence split was chosen as a low-performing open-domain evidence split that could provide interesting failures. Next, since the FinQA results were mostly equivalent, the *Whole-Document* evidence split was chosen as the main focus for a more distracting, real-world QA setting.

5.1.2. Prevalence & Pestilence of Common Reasoning Failures

Based on human qualitative analysis of random reasoning traces, three reasoning failures were arbitrarily identified to be investigated further: scoping errors, arithmetic errors, and looping errors. Figure 5.1 contains examples of arithmetic and looping errors while Figure 1.2 and Table 3.1 show examples of scoping errors. These failures were chosen, based on analysis of reasoning traces, for several reasons: looping errors accounted for most of the invalid reasoning traces, scoping errors were found most frequently, and arithmetic errors were chosen for their perceived importance in numerical settings.

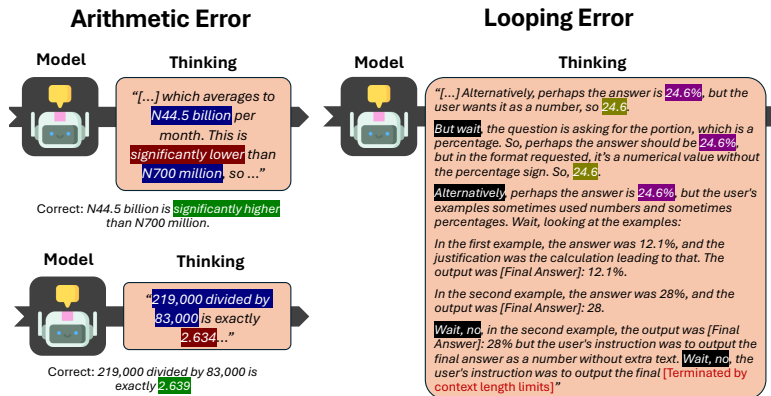


Figure 5.1: Examples of arithmetic and looping reasoning failures in LRM reasoning traces on numerical tasks.

The prevalence, how often they occurred, and pestilence, how often they contributed to an incorrect verdict, of each of these failures based on human annotation of 79 Distill-Qwen-generated reasoning traces is in Table 5.3. Due to time constraints, this analysis was limited to the *Re-Ranked* & *Whole-Document* evidence splits for QUANTEMP and FinQA, respectively. The QUANTEMP *Gold* evidence split was later annotated for scoping errors only and is included in the table for comparison.

The human annotations show that numerical scoping errors are the only failure type of the three that is both prevalent across tasks and often associated with incorrect outputs, appearing in 44.7% of QUANTEMP *Re-Ranked* traces and 26.8% of FinQA *Whole-Document* traces. In contrast, arithmetic and looping errors are less consistently prevalent: arithmetic errors are rare in QUANTEMP but more common in FinQA, while looping errors are highly pestilent when they occur but comparatively infrequent, especially in QUANTEMP. This suggests that scoping errors are the most systematic cross-task reasoning failure among the three error types considered. However, these annotations are limited to very few reasoning traces. To measure these errors at scale, an automatic judge was implemented.

Dataset	Evidence Category	Error Type	Correct Verdict		Incorrect Verdict		Total			
			Count	%	Count	%	Count	%	Contributed to Incorrectness	
									Count	%
QUANTEMP	Gold	Total Count	23		15		38			
		Numerical Scoping	8	53.3	6	40.0	14	36.8	4	28.6
	Re-Ranked	Total Count	19		19		38			
		Numerical Scoping	8	42.1	9	47.4	17	44.7	6	35.3
		Arithmetic	2	10.5	0	0.0	2	10.5	0	0.0
	Looping	0	0.0	1	6.7	1	2.6	1	100.0	
FinQA	Whole-Document	Total Count	26		15		41			
		Numerical Scoping	1	3.8	10	66.7	11	26.8	9	81.8
		Arithmetic	2	7.7	8	53.3	10	24.4	6	60.0
		Looping	0	0.0	5	33.3	5	12.2	5	100.0

Table 5.3: Reasoning failure prevalence as judged by a human annotator on 79 random reasoning traces generated by Deepseek-Distill-Qwen-7B on 38 fact-checking and 41 question-answering queries.

Table 5.4 contains the prevalence and pestilence of scoping errors, arithmetic errors, and looping errors as estimated by a straight LLM-as-a-Judge which is provided the ground truth answer on all evidence splits of the datasets.

The automatic judge broadly reinforces the human analysis by identifying numerical scoping errors as common and highly pestilent across both tasks, especially in incorrect verdicts. However, it also challenges the human pattern by marking arithmetic errors as extremely prevalent in FinQA, suggesting that arithmetic failures may be more task-dependent and abundant in computational tasks than first thought. Because the judge finds very few errors in correct outputs, these estimates should be interpreted cautiously as conflation and deflation of error detection is likely happening due to over-reliance on the final correctness label.

Dataset	Evidence Category	Error Type	Correct Verdict		Incorrect Verdict		Total			
			Count	%	Count	%	Count	%	Contributed to Incorrectness	
									Count	%
QUANTEMP	Gold	Total Count	1767		728		2495			
		Numerical Scoping	22	1.2	355	48.8	377	15.1	348	92.3
		Arithmetic	3	0.2	1	0.1	4	0.2	0	0.0
		Looping	0	0.0	2	0.3	2	0.1	2	100.0
	Qrels	Total Count	903		1592		2495			
		Numerical Scoping	33	3.7	632	39.7	665	26.7	626	94.1
		Arithmetic	5	0.6	2	0.1	7	0.3	2	28.6
		Looping	0	0.0	1	0.1	1	0.04	1	100.0
	Filtered Search Results	Total Count	919		1575		2494			
		Numerical Scoping	27	2.9	677	43.0	704	28.2	675	95.9
		Arithmetic	4	0.4	3	0.2	7	0.3	2	28.6
		Looping	1	0.1	0	0.0	1	0.04	1	100.0
Re-Ranked	Total Count	929		1566		2495				
	Numerical Scoping	45	4.8	658	42.0	703	28.2	655	93.2	
	Arithmetic	5	0.5	2	0.1	7	0.3	2	28.6	
	Looping	0	0.0	1	0.1	1	0.04	1	100.0	
FinQA	Gold	Total Count	446		700		1146			
		Numerical Scoping	1	0.2	183	26.1	184	16.1	182	98.9
		Arithmetic	1	0.2	328	46.9	329	28.7	320	97.3
		Looping	18	4.0	29	4.1	47	4.1	28	59.6
	Re-Ranked	Total Count	465		680		1145			
		Numerical Scoping	4	0.9	191	28.1	195	17.0	189	96.9
		Arithmetic	0	0.0	326	47.9	326	28.5	317	97.2
		Looping	13	2.8	37	5.4	50	4.4	26	72.0
	Whole-Document	Total Count	475		670		1145			
		Numerical Scoping	5	1.1	218	32.5	223	19.5	218	97.8
		Arithmetic	0	0.0	260	38.8	260	22.7	244	93.8
		Looping	6	1.3	13	1.9	19	1.2	13	68.4

Table 5.4: Reasoning failure prevalence as judged by a straight LLM-as-a-Judge (Qwen2.5-14B) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B. Occasionally, reasoning trace outcomes were invalid (<0.25% of traces), resulting in exclusion.

The straight LLM-as-a-Judge labels were compared to the gold human annotations for scoping errors, arithmetic errors, and looping errors. Table 5.5 shows that the judge only partially recovers the human-identified failures, with low recall for all three error types in both QUANTEMP and FinQA. In general, the straight LLM-as-a-Judge appears to underestimate the prevalence and overestimate the pestilence of various reasoning failures by searching more aggressively for failures when the final verdict is incorrect. This limitation is particularly important for scoping errors, which often interact with retrieval, interpretation, and final-answer correctness rather than appearing as isolated surface-level mistakes. The straight judge is therefore useful as a coarse indicator of failure patterns, but insufficient as a definitive measurement instrument for complex reasoning failures.

Dataset	Evidence Category	Error Type	Errors Correctly Identified		Errors Incorrectly Identified					Prediction Type	Exact Match Accuracy
			Count	%	Missing		Wrong Topic		Excess		
					Count	%	Count	%			
QUANTEMP	Re-Ranked	Numerical Scoping ($n=17$)	7	41.2	9	52.9	4	23.5	1	Total ($n=38$)	52.6
		Arithmetic ($n=2$)	0	0.0	5	100.0	0	0.0	0	Correct ($n=19$)	52.6
		Looping ($n=0$)	0		0		0		0	Incorrect ($n=19$)	52.6
FinQA	Whole-Document	Numerical Scoping ($n=11$)	3	27.3	8	72.7	0	0.0	1	Total ($n=41$)	56.1
		Arithmetic ($n=10$)	2	20.0	4	40.0	5	50.0	9	Correct ($n=26$)	67.7
		Looping ($n=5$)	1	20.0	4	80.0	0	0.0	0	Incorrect ($n=15$)	25.0

Table 5.5: Manual evaluation of straight LLM-as-a-Judge by a human annotator on stratified samples of fact-checking ($n=38$) and question-answering ($n=41$) reasoning traces generated by Deepseek-Distill-Qwen-7B and judgements generated by Qwen2.5-14B-Instruct.

Taken together, the human annotations and automatic measurements motivate focusing on numerical scoping errors as the central failure mode in this work. Scoping errors appear across numerical reasoning tasks and evidence settings, whereas arithmetic errors are mainly prominent in numerical QA and looping errors appear less prevalent overall and more tied to invalid or degenerate prompting behavior. Therefore, subsequent analyses focus on scoping errors as the most pervasive and task-general reasoning failure observed.

5.1.3. Taxonomizing Numerical Scoping Failures

Through an in-depth literature analysis of reasoning failure typologies, detailed in Sections 2.3 and 3.2, numerical scoping failures were positioned as logical problem representation failures.

Human qualitative analysis of generated reasoning traces from Table 5.3 further highlights a taxonomy of scoping errors, distinguishing extractive scoping errors, extrapolative scoping errors, and parametric scoping errors, as detailed in Section 3.2.2. A scoping error is categorized as extractive, extrapolative, or parametric depending on the type of the statistic that is being mis-scoped, as detailed in Section 3.2.2. For example, a statistic that is sourced from the evidence is an extractive statistic. If this statistic is assigned an incorrect unit, this is an extractive scoping error, because the mistake happened when *extracting* the statistic from evidence. An extrapolative statistic is mis-scoped if a mistake happened when *extrapolating* it from other statistics: for example, if statistics are combined improperly due to different time frames, making the current statistic unusable. A parametric statistic is mis-scoped if a mistake happened when naming it from *parametric* memory: for example, if a specific statistic is claimed from memory without proper grounding in the current setting. Table 3.1 contains representative examples of each type of scoping error.

These types of scoping failures outline where an error was made, allowing the development of specific practices to identify these errors, thus identifying scoping failures, a pervasive reasoning failure in RAG-based numerical reasoning.

RQ1: Key Insights

1. A common reasoning failure in RAG-based numerical reasoning is **numerical scoping errors**, which occur when a quantity is bound to incorrect referents. These failures are both prevalent and lead to incorrect verdicts across numerical fact-checking and QA tasks.
2. Numerical scoping errors can be categorized into **extractive**, **extrapolative**, and **parametric** variants depending on the source of the statistic that is being mis-scoped.
3. Straight LLM-as-a-Judge methods are sub-optimal when measuring numerical scoping errors, often leading to prevalence underestimation and pestilence overestimation.

5.2. RQ2: Measuring Numerical Scoping Failures in RAG-based Numerical Reasoning

Following the development of the taxonomy that categorizes numerical scoping failures into extractive, extrapolative, and parametric, the hybrid LLM-as-a-Judge pipeline was developed to detect scoping errors in reasoning traces at a statistic- and trace-level. Table 5.6 contains the overview of trace- and statistic- level scoping errors found across datasets and evidence splits.

Table 5.6 shows that trace-level scoping errors remain common across both tasks, but are more prevalent in QUANTEMP than in FinQA, consistent with the earlier human analysis. Scoping errors are generally more frequent in incorrect and invalid traces than in correct traces, although this pattern is not uniform across all evidence splits, indicating that scoping failures are associated with degraded performance but do not fully determine final correctness. The statistic-level counts also show that FinQA traces contain substantially more quantities on average, but a lower percentage of these quantities are mis-scoped. This suggests that QA reasoning is more quantity-dense while fact-checking reasoning about quantities is more vulnerable to scoping misalignment. This idea is quite natural as the QA task requires computing and outputting a number as the final answer, naturally requiring more quantities, while fact-checking requires careful reasoning over some quantities likely requiring more natural language.

Dataset	Evidence Category	Trace Correctness	% with Scoping Error	Average # of Quantities in Thinking Trace		
				Total	% Important	% Mis-Scoped
QUANTEMP	<i>Gold</i>	<i>All</i>	33.99	4.86	65.64	9.67
		Correct	33.28	4.72	65.68	9.96
		Incorrect	35.64	5.02	65.94	9.56
		Invalid	36.67	8.90	62.92	5.96
	<i>Qrels</i>	<i>All</i>	39.23	8.30	54.70	8.19
		Correct	36.90	8.38	55.01	7.28
		Incorrect	40.50	8.24	54.61	8.74
		Invalid	46.67	10.13	47.38	8.59
	<i>Filtered Search Results</i>	<i>All</i>	50.38	11.36	55.11	9.86
		Correct	48.68	11.34	55.20	9.79
		Incorrect	51.41	11.32	55.11	9.88
		Invalid	47.83	15.13	47.39	6.87
	<i>Re-Ranked</i>	<i>All</i>	34.62	5.38	60.78	9.11
		Correct	36.00	5.48	60.58	9.49
		Incorrect	33.87	5.33	60.79	9.01
		Invalid	22.22	3.67	57.49	8.99
FinQA	<i>Gold</i>	<i>All</i>	27.49	12.87	45.77	4.58
		Correct	21.14	11.32	43.99	3.27
		Incorrect	29.24	10.61	50.52	5.09
		Invalid	37.58	17.75	45.41	5.80
	<i>Re-Ranked</i>	<i>All</i>	30.63	12.71	46.34	4.96
		Correct	24.56	11.33	46.34	4.06
		Incorrect	27.56	10.27	47.22	4.58
		Invalid	43.56	17.02	45.83	6.17
	<i>Whole-Document</i>	<i>All</i>	29.23	15.36	44.08	3.97
		Correct	21.94	13.41	44.00	2.54
		Incorrect	26.60	12.96	45.14	4.01
		Invalid	48.31	22.25	43.51	5.93

Table 5.6: Scoping failure prevalence at a trace and statistic- level as judged by a hybrid LLM-as-a-Judge pipeline (Qwen2.5-72B) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-72B with different evidence splits. Rarely, judge outcomes were invalid (<0.8% of traces), resulting in exclusion.

Table 5.7 contains the specific statistic-level results including statistic classification, verification, and weighting, across datasets and evidence splits. The table shows that the type of statistic used in reasoning differs by task and evidence setting. In QUANTEMP, *Gold* evidence produces mostly extractive quantities, while open-domain evidence introduces many more parametric quantities, suggesting that poorer retrieval contexts encourage the model to rely more heavily on unsupported numerical knowledge. In contrast, FinQA contains a more balanced mixture of extractive, extrapolative, and parametric

quantities, reflecting the more calculation-heavy structure of the task. For QUANTEMP in general, scoping errors seem to occur wherever there are quantities; the distribution of scoping errors follows the distribution of statistics for each evidence split. In FinQA, unlike in QUANTEMP, although the distribution of statistics indicates extractive statistics as the most common, the most scoping errors are usually found in extrapolative or parametric statistics, suggesting a shift in reasoning focus away from extractive statistics. This is likely because of the relevant nature of evidence in FinQA where most statistics are clearly relevant or irrelevant, even in the more verbose *Whole-Document* setting.

Dataset	Evidence Category	Average # of Quantities in Thinking Trace										
								With Scoping Error				
		Total	% Important	% Extractive	% Extrapolative	% Parametric	% Unknown	Total	% Extractive	% Extrapolative	% Parametric	% Unknown
QUANTEMP	<i>Gold</i>	4.86	72.64	86.26	2.70	11.05	0.00	0.79	76.99	5.67	17.34	0.00
	<i>Qrels</i>	8.30	59.71	47.54	2.60	49.77	0.09	1.51	59.34	5.50	35.17	0.00
	<i>Filtered search results</i>	11.36	58.39	41.56	2.28	56.00	0.17	2.56	55.98	4.68	39.64	0.00
	<i>Re-Ranked</i>	5.38	67.31	66.00	3.33	30.64	0.03	0.91	57.42	8.59	33.99	0.00
FinQA	<i>Gold</i>	12.87	48.61	37.31	27.13	28.33	7.23	1.98	28.98	39.45	31.58	0.00
	<i>Re-Ranked</i>	12.71	49.38	38.23	27.29	27.97	6.52	2.06	34.61	34.44	30.95	0.00
	<i>Whole-Document</i>	15.36	46.92	38.28	28.32	30.29	3.12	2.55	28.51	34.15	37.34	0.00

Table 5.7: Statistic-level failure prevalence for extractive, extrapolative, and parametric scoping errors as judged by a hybrid LLM-as-a-Judge pipeline (Qwen2.5-14B) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B with different evidence splits. Rarely, judge outcomes were invalid (<0.8% of traces), resulting in exclusion.

The hybrid LLM-as-a-Judge pipeline was evaluated by comparing its statistic- and trace-level scoping error labels to gold human-annotated labels. Table 5.8 contains the results from this comparison. Due to time constraints this analysis was only done with *Gold* and *Whole-Document* evidence splits for QUANTEMP and FinQA, respectively.

The hybrid pipeline achieves moderate trace-level agreement with the human scoping labels, with weighted F1 scores of 57.9 for QUANTEMP and 64.7 for FinQA. This is not necessarily unexpectedly low, since the task of detecting scoping errors requires several difficult sub-questions: identifying all relevant quantities, classifying their source as extractive, extrapolative, or parametric, verifying whether the quantity is correctly scoped, and then aggregating these judgements to the trace level with statistic weighting. The statistic-level evaluation suggests that the main bottleneck is not quantity extraction, which is near-perfect, but rather the more semantic stages of classification and verification. Therefore, the hybrid judge should be treated as a scalable but imperfect measurement instrument. See Appendix Section B.2 for real scoping detection examples which explore difficulties in automatic detection.

Dataset	Reasoning Traces		Average Quantities Extracted Per Reasoning Trace			
	Total	Scoping Label W-F1	Total	% Correctly Identified	% Correctly Classified	% Correctly Verified
QUANTEMP	38	57.9	5.61	99.1	87.5	78.3
FinQA	41	64.7	17	99.4	75.7	76.8

Table 5.8: Manual evaluation of hybrid LLM-as-a-Judge scoping error detection pipeline by a human annotator on stratified samples of fact-checking (n=38) and question-answering (n=41) reasoning traces generated by Deepseek-Distill-Qwen-7B and judgements generated by Qwen2.5-14B-Instruct.

5.2.1. Distilling the Hybrid LLM-as-a-Judge Pipeline Signal

The hybrid LLM-as-a-Judge pipeline involves multiple expensive judge calls and CPU-intensive statistic extraction processes. Thus, several variants of student models were trained to recognize scoping errors based on the hybrid judge pipeline’s labels. Table 5.9 contains the relative speedup and agreement results between the student models and the hybrid judge pipeline. Student models are further evaluated by comparing downstream performance with test-time scaling techniques in Section 5.3.

The student models are substantially faster than the hybrid pipeline, with an average relative speedup of $212x$ across evidence splits. There is no meaningful difference between the relative speeds of the single- and multi-task ORMs, which follows naturally from their model architectures, since the multi-task ORM only adds one additional linear layer compared to the single-task ORM. Differences in relative speed across evidence splits likely reflect variation in hybrid-pipeline runtime caused by evidence length, number of extracted quantities, and run-to-run system variability such as background processes,

GPU scheduling, and I/O overhead.

The student models recover the hybrid judge signal conservatively, with stronger agreement and detection rates in *FinQA* than *QUANTEMP*, suggesting that scoping-error detection is more learnable in the more structured financial QA setting than in open-domain fact-checking. This result also follows from the statistic scoping error distributions from Table 5.7, which indicate extractive scoping errors are far less dominant in *FinQA* reasoning traces. Student models are only provided with the query and reasoning trace to judge without the input evidence, making detecting extractive scoping errors very difficult. Thus, *QUANTEMP* traces are likely harder to judge, given the relative abundance of extractive scoping errors compared to *FinQA*. The ablation of providing evidence to the ORMs is explored in Appendix B.

In general, the single-task ORM is as good or slightly better than the multi-task ORM in teacher agreement, indicating that adding a correctness objective does not clearly improve scoping-signal distillation, instead potentially inhibiting it slightly. Although *FinQA* distillation tends to perform slightly better, human agreement remains limited and mixed, therefore these ORMs should be interpreted as conservative, efficient proxies for the hybrid judge rather than validated replacements for human scoping annotation.

Dataset	Evidence Category	Student Model	Relative Speed	Agreement with Teacher (Hybrid Judge)				Agreement with Human Annotator	
				Teacher Scoping Rate	Student Scoping Rate	ROC-AUC	Label Agreement	ROC-AUC	Label Agreement
QUANTEMP	Gold	Single-Task Scoping ORM	113x		8.0	61.0	66.4	-	-
		Multi-Task Scoping (+ Correctness) ORM	111x	34.0	5.7	59.5	65.6	-	-
	Qrels	Single-Task Scoping ORM	143x		9.5	60.5	62.5	-	-
		Multi-Task Scoping (+ Correctness) ORM	143x	39.2	8.1	60.5	62.1	-	-
	Filtered Search Results	Single-Task Scoping ORM	192x		9.3	58.9	52.3	-	-
		Multi-Task Scoping (+ Correctness) ORM	192x	50.6	8.8	57.8	52.4	-	-
Re-Ranked	Baseline (Majority Class)	-		0.0	50.0	65.4	50.0	60.5	
	Single-Task Scoping ORM	110x	34.6	6.8	60.8	65.5	55.1	60.5	
	Multi-Task Scoping (+ Correctness) ORM	108x		6.5	60.4	65.9	64.9	60.5	
FinQA	Gold	Single-Task Scoping ORM	176x		11.3	70.7	72.3	-	-
		Multi-Task Scoping (+ Correctness) ORM	195x	27.5	14.5	68.1	72.2	-	-
	Re-Ranked	Single-Task Scoping ORM	439x		10.6	71.9	70.4	-	-
		Multi-Task Scoping (+ Correctness) ORM	438x	30.6	15.8	69.9	69.8	-	-
	Whole-Document	Baseline (Majority Class)	-		0.0	50.0	70.8	50.0	73.2
		Single-Task Scoping ORM	289x	29.2	17.4	75.1	72.4	65.8	78.0
Multi-Task Scoping (+ Correctness) ORM		319x		20.9	72.1	72.0	64.2	75.6	

Table 5.9: Evaluation of student models taught by the hybrid LLM-as-a-Judge scoping detection pipeline. Agreement is calculated between teacher scoping-error labels for whole test sets of *QUANTEMP* and *FinQA* and stratified samples of fact-checking ($n=38$) and question-answering ($n=41$) labels annotated by a human. Relative speed is calculated on average between the student and teacher models on the whole test sets. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct. The correctness head of the multi-task ORM is not evaluated.

RQ2: Key Insights

1. Given structure and specific measurable insights, **complex numeric scoping errors can be moderately measured** with a structured, hybrid LLM-as-a-Judge pipeline.
2. The scoping error signal from the hybrid judge pipeline can be moderately distilled into much smaller student models with moderate human annotation agreement and sizable speedups over the hybrid pipeline.

5.3. RQ3: Mitigating Numerical Scoping Failures & Effects on Downstream Performance

Scoping-aware parallel test-time scaling techniques (Section 4.6.4) were developed and leveraged to explore the impact of such techniques on downstream fact-checking and numerical QA performance.

5.3.1. Candidate Trace Generation

In order to perform parallel test-time scaling, K reasoning traces were generated for each *QUANTEMP* and *FinQA* test set query with varying temperatures. Due to computational constraints, only certain evidence splits were used following the reasoning from Section 5.1.1: *Gold* & *Re-Ranked* for *QUANTEMP* and *Whole-Document* for *FinQA*. The downstream performance of these candidate reasoning traces across temperatures, generation models, and K values are presented in Tables 5.10 and 5.11.

Across candidate traces, the two generation backbones show different strengths across tasks. On *QUANTEMP*, Deepseek-Distill-Llama-8B consistently outperforms Deepseek-Distill-Qwen-7B, improv-

ing weighted-F1 by roughly five points in the *Gold* evidence setting and more than ten points in the more difficult *Re-Ranked* setting. In contrast, on FinQA, Deepseek-Distill-Qwen-7B achieves higher exact match accuracy across all temperatures. The Deepseek-Distill-Qwen-7B model was created from a Qwen-2.5-Math-7B model, which focuses on math and logic capabilities [13, 63, 64]. Meanwhile, the Deepseek-Distill-Llama-8B model was created from a general purpose Meta Llama-3.1-8B-Base model [13, 18]. These differences in numerical FC and QA performance may be partly explained by the alignment between each backbone’s pretraining and the demands of each task. Overall, no single inference model is uniformly stronger across the tasks in this work.

Temperature effects are relatively modest, but lower temperatures generally produced stronger individual trace performance for QUANTEMP with *Gold* evidence. In contrast, QUANTEMP with *Re-Ranked* evidence and FinQA with *Whole-Document* evidence show little systematic temperature effect. These results motivate using multiple candidate traces across temperatures for test-time scaling, since temperature variation introduces some answer diversity without dramatically degrading average trace quality. Overall, both generation models produce sufficiently valid and diverse candidate traces to support scoping-aware test-time scaling.

Dataset	Evidence Category	Temperature	Distill-Qwen					Distill-Llama				
			Per-Class			Total		Per-Class			Total	
			T-F1	F-F1	C-F1	M-F1	W-F1	T-F1	F-F1	C-F1	M-F1	W-F1
QUANTEMP	<i>Gold</i>	0.2	73.03	81.74	47.97	67.58	72.38	77.40	85.96	56.63	73.33	77.31
		0.3	70.45	81.41	46.39	66.08	71.31	76.30	85.52	54.71	72.18	76.39
		0.4	70.46	80.79	46.80	66.02	70.90	76.88	85.49	55.47	72.62	76.67
		0.5	70.35	80.75	46.17	65.76	70.83	76.62	86.04	55.61	72.76	76.97
	<i>Re-Ranked</i>	0.2	35.40	38.40	34.09	35.96	36.79	38.59	56.41	35.40	43.47	48.00
		0.3	36.51	38.46	35.12	36.70	37.29	40.05	56.58	33.94	43.53	48.02
		0.4	36.29	39.26	34.45	36.67	37.55	38.21	56.90	34.94	43.35	48.08
		0.5	37.08	38.61	34.35	36.68	37.30	38.47	56.13	34.57	43.06	47.61

Table 5.10: Average fact-checking outcome-level performance across generation temperatures for $K = 12$ candidate traces ($k = 3$ per temperature) on the QUANTEMP test set ($n=2,495$) with *Gold* & *Re-Ranked* evidence. Traces were generated from either Deepseek-Distill-Qwen-7B or Deepseek-Distill-Llama-8B. Reasoning trace outcomes were sometimes invalid (0.4-0.6% of Distill-Qwen traces; 0.01-0.1% of Distill-Llama traces), resulting in exclusion.

Dataset	Evidence Category	Temperature	Exact Match Accuracy	
			Distill-Qwen	Distill-Llama
FinQA	<i>Whole-Document</i>	0.2	67.30	61.75
		0.3	67.24	62.04
		0.4	67.11	62.62
		0.5	65.28	62.50

Table 5.11: Average question-answering outcome-level performance across generation temperatures for $K = 20$ candidate traces ($k = 5$ per temperature) on the FinQA test set ($n=1,147$) with *Whole-Document* evidence. Traces were generated from either Deepseek-Distill-Qwen-7B or Deepseek-Distill-Llama-8B. Reasoning trace outcomes were sometimes invalid (24.9-28.5% of Distill-Qwen traces; 7.1-8.2% of Distill-Llama traces), resulting in exclusion.

5.3.2. Impacts of Test-Time Scaling on Downstream Performance

Table 5.12 contains the test-time scaling downstream performances of the different scoping error mitigation variants described in Sections 3.4 and 4.6.4. Learning curves and ablations for training the learned reward model variants across datasets can be seen in Appendix B. Appendix tables B.5 and B.6 contain real examples of candidate traces chosen by scoping-aware mitigation variants.

Table 5.12 shows that test-time scaling improves performance most clearly when answer selection is guided by a correctness signal, with smaller gains from adding scoping-aware signals in certain contexts. Overall, scoping-aware mitigation can produce small improvements over non-scoping baselines, but no large, uniform gains. Furthermore, the findings mostly generalize over different inference models, demonstrating that these trends are not artifacts of single generation model families.

The upper bound results indicate that scoping filtering removes some correct candidate answers, since

the scoping-filtered oracle is consistently lower than the unconstrained oracle across evidence settings. However, the gap is not catastrophic for most evidence settings: the filtered oracle remains high, especially for QUANTEMP *Gold* and FinQA *Whole-Document*, each with a gap of around 3 points, suggesting that many correct answers remain available after removing traces judged to contain scoping errors. However, for QUANTEMP *Re-Ranked*, the performances were reduced from 76.74 to 69.60 for Distill-Qwen and 79.64 to 74.20 for Distill-Llama, quite sizable reductions. This suggests that the hybrid pipeline scoping filtering may be too harsh in chaotic evidence settings with many concurrent errors. Overall, scoping filtering retains many correct traces, but also can sometimes filter a little too excessively, making conservative alternatives more favorable.

For QUANTEMP with *Gold* evidence, self-consistency is the strongest non-oracle strategy, outperforming all ORM-based selection variants. This likely reflects the strength of the evidence setting: when human fact-checker evidence already contains the relevant grounding information and final verdict, the majority answer across sampled traces is often sufficient. In this setting, scoping filtering slightly reduces self-consistency performance, suggesting that filtering can discard useful traces when the evidence is already highly reliable.

In contrast, scoping filtering improves self-consistency in the Distill-Qwen harder evidence settings: QUANTEMP *Re-Ranked* improves from 35.42 to 36.84 weighted-F1, and FinQA *Whole-Document* improves from 68.53 to 69.01 exact match accuracy. These gains are modest, but they suggest that scoping errors become more useful as a mitigation target when evidence is noisier, longer, or less directly aligned with the answer for Distill-Qwen traces. This pattern is consistent with the idea that scoping failures are most harmful when the model must decide which quantities in the context are actually relevant. However, this pattern does not hold for Distill-Llama, limiting its impact.

Correctness-based selection is also affected by scoping-aware filtering, but not uniformly. In the *Gold* and *Whole-Document* settings, adding scoping filtering to the correctness ORM improves performance across inference models. Scoping filtering using the student single-task scoping ORM rather than the hybrid judge pipeline produces even stronger gains for FinQA *Whole-Document* when paired with the correctness ORM. However, in QUANTEMP *Re-Ranked*, scoping filtering hurts the correctness ORM, suggesting that the filtering signal may remove traces that are still answer-correct under noisy open-domain evidence, or that the scoping detector is less reliable in this setting.

Dataset	Evidence Category	Type of Mitigation	Mitigation Strategy	Weighted-F1	
				Distill-Qwen	Distill-Llama
QUANTEMP	<i>Gold</i>	Upper Bound	Oracle	93.97	93.94
		Upper Bound (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Oracle	90.52	91.29
		Baseline	Self-Consistency	75.03	80.14
		Candidate (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Self-Consistency	74.68	79.17
		Baseline	Correctness ORM	72.06	77.49
		Candidate (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Correctness ORM: Correctness Best-of-N	73.05	78.47
	<i>Re-Ranked</i>	Candidate (Scoping-Aware)	Scoping ORM: Scoping Filtering + Correctness ORM: Correctness Best-of-N	72.06	77.30
		Candidate (Scoping-Aware)	Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	73.30	76.00
		Candidate (Scoping-Aware)	Multi-Task ORM: Scoping + Correctness Best-of-N	73.07	76.40
		Upper Bound	Oracle	76.74	79.64
		Upper Bound (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Oracle	69.60	74.20
		Baseline	Self-Consistency	35.42	49.84
		Candidate (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Self-Consistency	36.84	49.25
		Candidate (Scoping-Aware)	Correctness ORM: Correctness Best-of-N	55.54	58.81
Candidate (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Correctness ORM: Correctness Best-of-N	52.73	58.13		
Candidate (Scoping-Aware)	Scoping ORM: Scoping Filtering + Correctness ORM: Correctness Best-of-N	54.79	58.49		
Candidate (Scoping-Aware)	Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	56.19	60.28		
Candidate (Scoping-Aware)	Multi-Task ORM: Scoping + Correctness Best-of-N	56.46	59.96		
FinQA	<i>Whole-Document</i>			Exact Match Accuracy	
				Distill-Qwen	Distill-Llama
		Upper Bound	Oracle	83.51	83.30
		Upper Bound (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Oracle	81.39	79.86
		Baseline	Self-Consistency	68.53	65.81
		Candidate (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Self-Consistency	69.01	65.66
		Baseline	Correctness ORM: Correctness Best-of-N	69.24	66.61
		Candidate (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Correctness ORM: Correctness Best-of-N	69.81	67.70
		Candidate (Scoping-Aware)	Scoping ORM: Scoping Filtering + Correctness ORM: Correctness Best-of-N	72.96	71.11
		Candidate (Scoping-Aware)	Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	74.20	73.06
		Candidate (Scoping-Aware)	Multi-Task ORM: Scoping + Correctness Best-of-N	68.71	66.70

Table 5.12: Downstream performance of different scoping error-aware test-time scaling strategies for $K = 12$ fact-checking candidate traces and $K = 20$ numerical QA candidate traces on the QUANTEMP ($n=2,495$) and FinQA ($n=1,147$) test sets. Traces were generated from either Deepseek-Distill-Qwen-7B or Deepseek-Distill-Llama-8B. Reasoning trace outcomes were sometimes invalid (0.0-2.1% of traces), resulting in exclusion.

The multi-task ORM variants are among the strongest learned strategies overall, although the best variant differs by evidence setting and inference model. The combined scoping-and-correctness-best-of- N variant performs best for Distill-Qwen QUANTEMP *Re-Ranked*, while the multi-task correctness best-of- N with scoping filtering performs best for FinQA *Whole-Document*, Distill-Llama QUANTEMP *Re-Ranked*, and Distill-Qwen QUANTEMP *Gold*. However, for Distill-Llama QUANTEMP *Gold*, the hybrid pipeline scoping filtering with the correctness ORM performs best among the ORM-based variants. The multi-task score is also sensitive to the weighting hyperparameter α , which controls the relative contribution of the scoping signal. The current value ($\alpha = 0.18$) performs well for QUANTEMP *Re-Ranked* but poorer for the other splits. This motivates the ablation of α in Appendix B, since the usefulness of soft scoping penalties appears to depend on the distribution of scoping errors and the reliability of the correctness signal. Overall, the multi-task ORM results suggest that modeling correctness and scoping jointly can improve trace selection, but that the optimal way to combine these signals depends on the task, evidence regime, and hyperparameters.

To examine the impact K has on mitigation variants, Figure 5.2 compares the downstream performances of mitigation variants across K values, generation models, and evidence splits. Across K values, the oracle upper bounds increase substantially as more candidate traces are generated, especially for QUANTEMP *Re-Ranked*, showing that additional traces often contain correct answers even when aggregation methods fail to select them. Furthermore, consistent with earlier results, the hybrid pipeline scoping filtering reduces this oracle performance slightly, but not substantially, meaning that scoping filtered traces still have space to be selected correctly for most evidence settings. However,

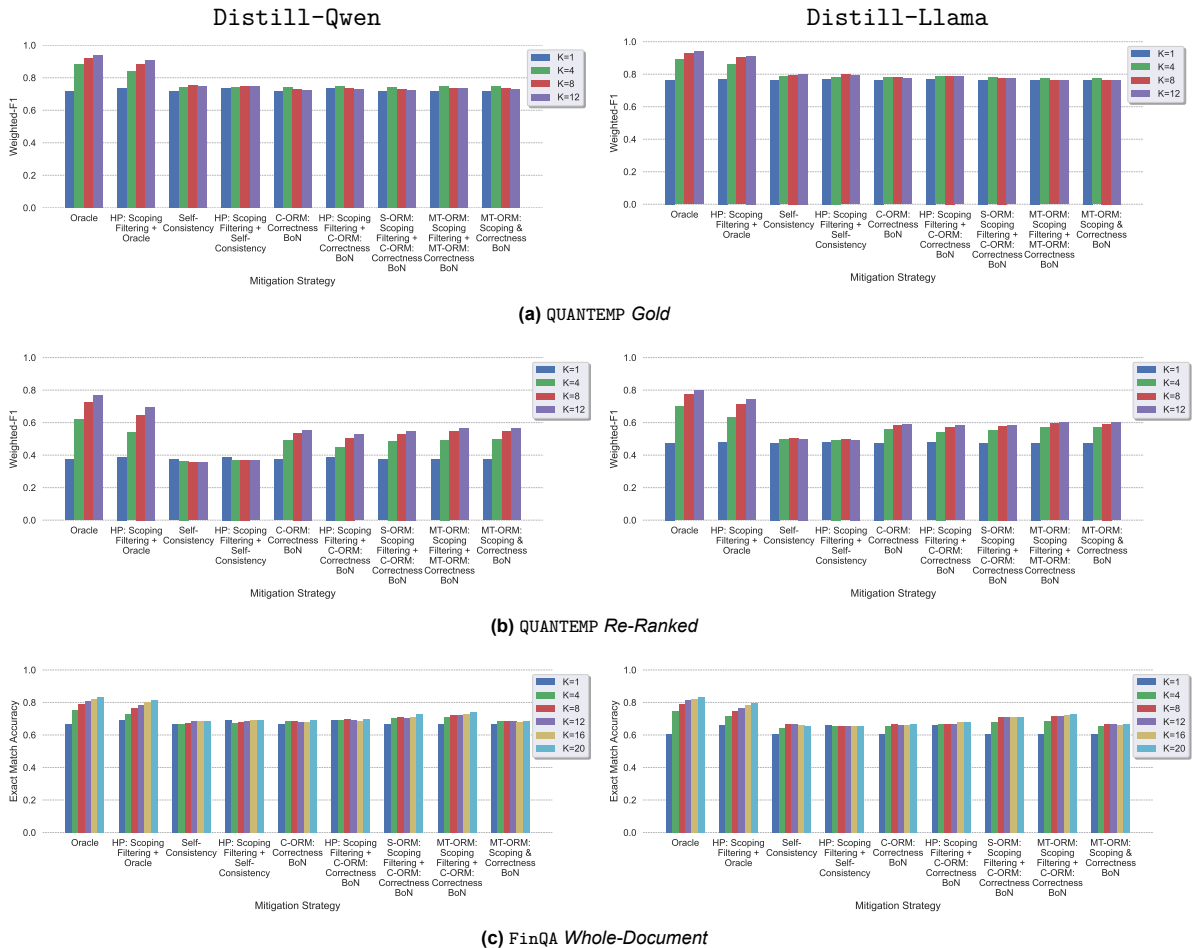


Figure 5.2: Downstream performance of scoping-aware and scoping-blind test-time scaling strategies across different numbers of candidate traces (K) from the Deepseek-Distill-Qwen-7B (left) and Deepseek-Distill-Llama-8B (right) inference models. Scoping-aware strategies include the Hybrid LLM-as-a-Judge Pipeline (HP), Scoping Outcome Reward Model (S-ORM), and Multi-Task ORM (MT-ORM). Scoping-blind strategies include Self-Consistency and the Correctness ORM (C-ORM). For QUANTEMP, performance is reported as weighted-F1; for FinQA, performance is reported as exact match accuracy.

the non-oracle mitigation strategies improve much less with increasing K , suggesting that reliable trace selection is a significant bottleneck. In *QUANTEMP Gold*, most non-oracle strategies remain relatively stagnant across K , consistent with earlier observation that self-consistency is already strong in this evidence setting without needing additional test-time scaling techniques. In contrast, the larger separation between oracle and learned strategies in *QUANTEMP Re-Ranked* and *FinQA Whole-Document* suggests that better selection mechanisms could still yield further gains. Both *QUANTEMP Re-Ranked* and *FinQA Whole-Document* showcase examples where scoping-aware reward signals slightly outperform their scoping-blind counterparts; and this pattern stays mostly consistent across K values. This encourages the idea that the scoping-specific downstream performance gains are not an artifact of specific K values, but an overall trend in open-domain evidence settings.

Taken together, these mitigation results show that scoping-aware mitigation is useful as a supplementary reward signal across test-time scaling techniques. Scoping signals can improve self-consistency and reward-model selection in several settings, and they appear especially helpful when combined with correctness-based selection, but the gains are modest and inconsistent. Furthermore, the consistency of these patterns across generation models and K -values suggest that the observed effects are not artifacts of a single generation model or specific K , but reflect a recurring, though limited, benefit of scoping-aware selection in open-domain numerical reasoning. The following analyses therefore examine potential limiting factors, including imperfect scoping detection, the relationship between scoping errors and final-answer correctness, and the effect of answer diversity on parallel test-time scaling.

Impact of Scoping-Aware Reward Signals

Scoping-aware reward signals do not always have significant impact on downstream performance compared to their scoping-blind counterparts. This work critically examines the potential contributors to this limited impact. First, the hybrid judge pipeline detects scoping errors imperfectly. Next, scoping errors may not always contribute to answer incorrectness. Third, the diversity of candidate traces may provide little room for test-time scaling to improve performance. Finally, scoping-blind mitigation techniques may not be as scoping blind as they first appear.

Sub-Optimal Scoping Error Detection

As seen in Table 5.8, the hybrid LLM-as-a-Judge pipeline used to introduce scoping error signals into the mitigation variants only moderately detects scoping errors at the trace level. This limits the effectiveness of scoping-aware mitigation in two directions: false negatives allow mis-scoped traces to remain in the candidate pool, while false positives remove traces that may still contain correct answers. As a result, scoping filtering is a noisy intervention rather than a clean removal of harmful reasoning traces, which likely contributes to the modest and inconsistent downstream gains observed in Table 5.12. This limitation is especially relevant for mitigation strategies that rely on hard filtering, since a single incorrect trace- or statistic-level scoping label can directly determine whether a candidate answer is retained or discarded. Also, this noisy signal is distilled into student verifiers which mimic this noise, making them less effective as well.

Non-Critical Scoping Errors to Trace Correctness

Furthermore, scoping errors may not always be critical to outcome correctness. Through human qualitative analysis, 67.7% of scoping errors in fact-checking and 18.2% in numerical question answering were identified as non-critical (Table 5.3). This means that removing traces with scoping errors may not improve correctness by a large margin. However, through statistical testing, the correctness of traces with and without scoping errors was compared and significant associations were found even with the noisy scoping labels of the hybrid pipeline in some evidence settings; these associations can be seen in Figure 5.3.

For *QUANTEMP Gold* and *FinQA Whole-Document*, across all tested K values, traces without scoping errors had a higher observed correctness rate than traces with scoping errors. Through clustered logistic regression via generalized estimating equations, clustered by query, scoping error presence was identified as a statistically significant negative predictor of correctness for all tested K values. The estimated odds ratios ranged from 0.58 to 0.89 with p-values of 3×10^{-7} to 0.044, indicating that traces containing scoping errors had significantly lower odds of being correct than traces without scoping errors, even after adjusting for within-question correlation, across K values. However, for *QUANTEMP*

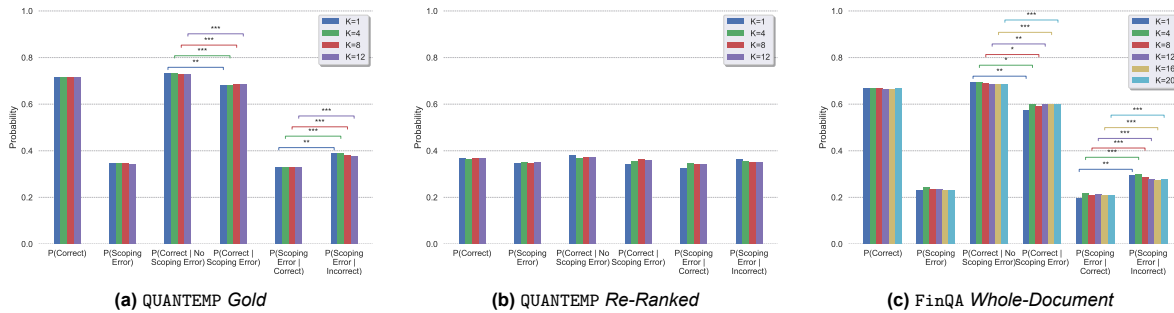


Figure 5.3: Marginalized and conditional probabilities of the co-occurrence of correctness and scoping errors across different numbers of *Distill-Qwen*-generated candidate traces (K) for QUANTEMP and FinQA test sets. Clustered logistic regression based on generalized estimating equations, clustered on query identity was used to determine statistical significance at a confidence level of 0.95. A statistical significance level of $p < 0.05$ is marked with *, $p < 0.01$ with **, and $p < 0.001$ with ***.

Re-Ranked, no such association was found with the odds ratios ranging from 0.84 to 0.95 with p-values of 0.05 to 0.17. Perhaps the lack of such associations in the *Re-Ranked* evidence setting suggests there are other reasoning failures that are larger limiting factors to answer correctness than scoping errors are, making the association between answer correctness and scoping error presence too noisy to be statistically significant in this evidence setting.

The inconsistency of this association suggests that scoping errors are meaningful indicators of reasoning quality, but they are not equivalent to answer incorrectness. Therefore, it makes sense that scoping-aware mitigation on its own has limited impact and can be most useful when combined with correctness-based selection rather than used as a standalone criterion.

Examining Candidate Trace Diversity

Next, test-time scaling can only improve downstream performance if the correct answer is actually generated in the candidate traces. The upper bound oracle results in Table 5.12 depict the maximum possible downstream performance in each evidence split and can only occur when the exact correct candidate trace is chosen for each query. Answer disagreement and entropy across evidence splits is depicted in Table 5.13.

Across settings, answer entropy and disagreement increase with larger K , indicating that additional samples do introduce more answer variation. However, the diversity remains moderate, and scoping filtering generally reduces entropy and disagreement by removing part of the candidate pool. For QUANTEMP *Gold*, even at $K = 12$, answer sets are still fairly concentrated rather than wildly spread out, with the majority label taking up about 84% of all valid answers and 87% of all non-scoping valid answers. For QUANTEMP *Re-Ranked* and FinQA *Whole-Document*, answers are slightly more diverse with 76% and 74% of answers taken up by the majority label at $K = 12$, respectively. In general, this limited candidate trace diversity may reduce the gains seen from parallel test-time scaling: if most traces converge on the same answer, or if filtering removes both erroneous and useful alternative answers, then aggregation has limited opportunity to recover from the initial model’s mistakes. Examples of these aggregation decisions can be seen in Appendix Tables B.5 and B.6.

Dataset	Evidence Category	Scoping Filtered?	Avg. Normalized Answer Entropy					Avg. Answer Disagreement				
			$K = 4$	$K = 8$	$K = 12$	$K = 16$	$K = 20$	$K = 4$	$K = 8$	$K = 12$	$K = 16$	$K = 20$
QUANTEMP	<i>Gold</i>	No	25.33	31.30	33.67	-	-	14.66	16.29	16.81	-	-
		Yes	17.08	24.77	27.75	-	-	10.57	13.54	14.43	-	-
	<i>Re-Ranked</i>	No	35.77	43.58	46.56	-	-	20.74	23.09	24.08	-	-
		Yes	25.25	34.96	39.29	-	-	15.82	19.37	20.99	-	-
FinQA	<i>Whole-Document</i>	No	43.53	54.03	56.22	56.59	56.39	19.79	24.65	26.47	24.75	28.19
		Yes	30.59	45.93	51.08	53.59	54.47	19.79	20.62	23.15	27.34	25.89

Table 5.13: Average answer entropy and disagreement across varying K values for QUANTEMP ($n = 2,495$) and FinQA ($n = 1,147$) test set *Distill-Qwen*-generated candidate traces with and without scoping errors. Reasoning trace outcomes were sometimes invalid (0.4-8.1% of traces), resulting in exclusion. Due to computational limits, QUANTEMP test-time scaling experiments were only run with a maximum K value of 12.

Implicit Scoping Error Awareness in Correctness Verifier Scores

Finally, as discussed in Section 3.2, reasoning failures are inter-related. Thus, it naturally follows that general correctness ORMs can also slightly measure and mitigate scoping errors indirectly. Through statistical testing, the correctness ORM verifier scores between traces with and without scoping errors were compared and some significant associations were found between scoping error presence and lower correctness ORM scores for some evidence splits and K values (Figure 5.4).

These associations were strongest in *FinQA Whole-Document*: across all tested K values, traces without scoping errors had higher observed correctness ORM scores than traces with scoping errors. Through clustered linear regression via generalized estimating equations, clustered by query, scoping error presence was identified as a statistically significant negative predictor of correctness ORM score for all tested K values. The estimated coefficients ranged from -0.058 to -0.030 , with p-values decreasing from 0.029 at $K = 1$ to 8×10^{-11} at $K = 20$, indicating that traces containing scoping errors received significantly lower correctness ORM scores, even after adjusting for within-question correlation, across K values.

However, for *QUANTEMP* this association was weaker and less consistent. In the *Gold* evidence setting, scoping error presence was a significant negative predictor at $K = 1$ and $K = 12$, with coefficients ranging from approximately -0.046 to -0.013 . However, for *QUANTEMP Re-Ranked*, no statistically significant association was found across tested K values. This may suggest that, in the noisier open-domain evidence setting, the learned correctness verifier is less focused on scoping errors as other reasoning or retrieval failures are larger limiting factors.

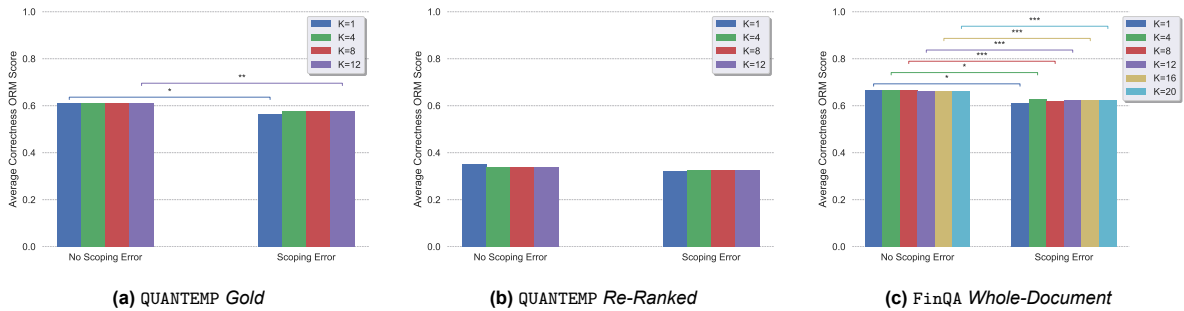


Figure 5.4: Average correctness outcome reward model (ORM) scores given the *QUANTEMP* or *FinQA Distill-Qwen*-generated trace did or did not have a detected scoping error according to the Hybrid LLM-as-a-Judge pipeline, across different numbers of candidate traces (K). Clustered linear regression based on generalized estimating equations, clustered on query identity was used to determine statistical significance at a confidence level of 0.95. A statistical significance level of $p < 0.05$ is marked with *, $p < 0.01$ with **, and $p < 0.001$ with ***.

Overall, this suggests that the correctness ORM is not fully scoping-blind: traces with scoping errors often receive lower correctness scores, even without an explicit scoping signal. This implicit sensitivity may explain why adding an explicit scoping signal sometimes produces only small improvements over the correctness ORM alone in Table 5.12.

5.3.3. Scoping-Aware Mitigation Overview

In general, scoping-aware mitigation techniques in test-time scaling slightly outperformed their not-so-scoping-aware counterparts, but not substantially or consistently. This pattern was observed across K values and both *Deepseek-Distill-Qwen-7B* and *Deepseek-Distill-Llama-8B* reasoning traces, suggesting that the benefit and limitations of scoping-aware selection are generalizable across K values and inference models. Several reasons for the limited impact of scoping-aware selection are explored: scoping error detection remains imperfect, scoping errors are not always critical to final-answer correctness, candidate traces may lack sufficient answer diversity, and correctness-based reward models may already capture some scoping-related failures indirectly. Overall, the distilled scoping-aware multi-task mitigation variants produced the strongest downstream performance in open-domain evidence settings, but these variants were sensitive to hyperparameters. Together, these limitations and results suggest that scoping-aware reward signals are useful indicators of reasoning quality but need other signals to consistently drive downstream improvements.

This work explored one sub-branch of potential mitigation techniques: post-hoc trace selection during parallel test-time scaling. The mitigation results suggest that scoping-aware signals can provide useful selection information, but that simple scoping-aware filtering or ranking is not sufficient on its own to consistently improve downstream performance. These findings point to the need for more robust scoping detection, stronger integration with correctness signals, and the exploration of other mitigation strategies that intervene beyond post-hoc candidate selection.

RQ3: Key Insights

1. Using scoping-aware reward signals to select between candidate reasoning traces **modestly improves downstream performance, but not uniformly** across inference models, evidence splits, K values, and mitigation techniques.
2. Distilled scoping-aware multi-task mitigation variants are often among the highest-performing approaches across evidence splits and inference models, but show sensitivity to hyperparameters.
3. The limiting factors for scoping-aware downstream performance improvements using test-time scaling include imperfect scoping error detection, non-critical scoping errors, limited candidate-answer diversity, and implicit scoping awareness in correctness reward models.

6

Conclusion

Numerical information plays a central role in how people interpret real-world events, evaluate claims, and make decisions. As LLM-based assistants and RAG systems are increasingly used for open-domain information seeking, their ability to reason reliably over quantities becomes critical. This thesis investigated one underexplored reason why such systems fail: numerical scoping errors, where a model incorrectly binds a quantity to the wrong referents.

To answer RQ1, this work identified numerical scoping errors as a common and consequential reasoning failure in open-domain numerical reasoning. Through human qualitative analysis of reasoning traces, scoping errors were found across both numerical fact-checking and numerical question-answering tasks. Compared to arithmetic and looping errors, scoping errors were the most task-general failure type observed: arithmetic errors were more concentrated in numerical QA, while looping errors were comparatively rare and tied to degenerate generations. This supports a central claim of this thesis: numerical reasoning failures are not commonly caused by an inability to calculate, but by an inability to correctly contextualize numbers.

To answer RQ2, this thesis developed a hybrid LLM-as-a-Judge pipeline for detecting scoping errors at the statistic- and trace-level. The pipeline decomposes reasoning traces into quantities, classifies those quantities as extractive, extrapolative, or parametric, verifies whether they are correctly scoped, and aggregates statistic-level judgements into trace-level labels. The results show that scoping errors can be moderately measured at scale: the hybrid pipeline achieved moderate agreement with human annotations, and distilled student models reproduced the hybrid signal conservatively while providing large speedups. These findings suggest that structured judging is a useful step beyond straight LLM-as-a-Judge prompting, but that automatic scoping detection remains a difficult measurement problem.

To answer RQ3, this work evaluated whether scoping-aware reward signals can improve downstream performance during parallel test-time scaling. Scoping-aware strategies produced modest improvements in some settings, especially when combined with correctness-based selection, but the gains were not large or uniform. Several factors were explored to explain this: scoping detection signals are noisy, some scoping errors are non-critical to final answer correctness, candidate traces often lack enough answer diversity, and correctness ORMs already implicitly capture some of the scoping signal. The highest-performing downstream strategies were most often the more conservative scoping-aware multi-task variants, which combine scoping and correctness objectives in one ORM. In general, scoping-aware reward signals were useful indicators of reasoning quality but needed other signals to drive downstream improvements.

Overall, this thesis demonstrates that fine-grained failure characterization is a useful methodology for studying open-domain numerical reasoning. Numerical scoping errors expose a gap between producing fluent reasoning traces and genuinely grounding numerical reasoning with correct context. In high-impact settings, a single mis-scoped number can change the meaning of a conclusion, turning a statistic into a misleading financial signal, an unsafe medical recommendation, or a false public claim. Improving open-domain numerical reasoning will therefore require systems to do more than retrieve

evidence, generate chain-of-thought, or select high-scoring outputs: they must also track what each number means, where it comes from, and under which conditions it is valid.

6.1. Limitations

This work has several limitations. First, the human annotation process was limited in scale and was primarily conducted by one annotator. Although the annotations were sufficient to motivate and evaluate the proposed scoping-error taxonomy, larger multi-annotator studies would be needed to establish stronger reliability and produce more robust estimates of scoping-error prevalence and pestilence.

Next, the hybrid LLM-as-a-Judge pipeline achieved only moderate agreement with human scoping labels, displaying a serious gap. This limitation is particularly impactful because the hybrid pipeline was used during scoping error mitigation, meaning that the downstream mitigation results are bound by the quality of this scoping detector as well.

Third, the datasets and evidence impose constraints on the conclusions. QUANTEMP provides a challenging open-domain fact-checking setting with noisy retrieved textual evidence and frequently co-occurring retrieval and reasoning failures. In contrast, FinQA contains open numerical answers computed from verbose textual and tabular evidence, where only small portions are relevant and the evidence remains document-bounded rather than truly open-domain. The two tasks provide complementary but incomplete coverage of real-world numerical reasoning, limiting the widespread generalizability of this work.

Finally, the mitigation experiments explored only a subset of possible design choices. The work focused on post-hoc trace selection during parallel test-time scaling, using specific candidate generation settings, reward model architectures, filtering thresholds, and score-combination hyperparameters. Although some ablations were performed, many hyperparameters and mitigation variants were not exhaustively tuned. The modest gains should therefore be interpreted as evidence about this mitigation family, not as a definitive upper bound on the usefulness of scoping-aware reasoning signals.

6.2. Future Work

Future work should first expand human analysis. A larger, multi-annotator study would make it possible to refine the scoping-error taxonomy, measure inter-annotator agreement, and more reliably estimate scoping error prevalence and which scoping errors are actually critical to final answer correctness.

Next, future work should improve automatic scoping-error detection. The hybrid pipeline showed that structured statistic-level judging is feasible, but classification and verification remain challenging. Better quantity extraction, alignment, understanding, classification, and verification with uncertainty estimates could improve the reliability of scoping labels. Stronger scoping detectors would also make downstream mitigation experiments more informative.

Third, mitigation should be explored beyond post-hoc trace selection. Scoping signals could be used earlier in the reasoning process through scoping-aware prompting, intermediate statistic-level trace repair, or generator fine-tuning. Rather than only filtering traces after generation, systems could explicitly check whether each quantity is bound to the correct referents before producing the final answer.

Finally, future work should evaluate scoping errors across more models and datasets. This thesis focused mainly on reasoning traces from Deepseek-Distill-Qwen-7B and Deepseek-Distill-Llama-8B. Extending the analysis to other reasoning and non-reasoning LLMs, as well as datasets such as TabFact [8] or TAT-QA [70], would clarify whether numerical scoping errors are a general property of open-domain numerical reasoning or depend strongly on model family, task format, and evidence type.

6.3. Disclosure

In line with TU Delft's publishing policies¹, I acknowledge the use of AI tools powered by ChatGPT to assist in rephrasing certain sections of this thesis. The icons featured in the figures were created by various contributors from www.flaticon.com. All AI-generated content has been thoroughly reviewed to ensure its accuracy.

¹TU Delft publishing policies: <https://www.tudelft.nl/library/actuele-themas/open-publishing/about/policies>

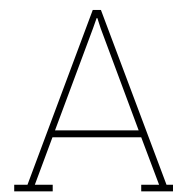
References

- [1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. <https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2>. 2024.
- [2] Garima Agrawal et al. *Mindful-RAG: A Study of Points of Failure in Retrieval Augmented Generation*. 2024. arXiv: 2407.12216 [cs.IR]. URL: <https://arxiv.org/abs/2407.12216>.
- [3] Aisha Alansari and Hamzah Luqman. *A Comprehensive Survey of Hallucination in Large Language Models: Causes, Detection, and Mitigation*. Oct. 2025. DOI: 10.48550/arXiv.2510.06265.
- [4] Yuntao Bai et al. *Constitutional AI: Harmlessness from AI Feedback*. 2022. arXiv: 2212.08073 [cs.CL]. URL: <https://arxiv.org/abs/2212.08073>.
- [5] Johan Boye and Birger Moell. *Large Language Models and Mathematical Reasoning Failures*. 2025. arXiv: 2502.11574 [cs.AI]. URL: <https://arxiv.org/abs/2502.11574>.
- [6] Andrew Brown, Muhammad Roman, and Barry Devereux. *A Systematic Literature Review of Retrieval-Augmented Generation: Techniques, Metrics, and Challenges*. 2025. arXiv: 2508.06401 [cs.DL]. URL: <https://arxiv.org/abs/2508.06401>.
- [7] Wenhui Chen et al. *Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks*. 2023. arXiv: 2211.12588 [cs.CL]. URL: <https://arxiv.org/abs/2211.12588>.
- [8] Wenhui Chen et al. *TabFact: A Large-scale Dataset for Table-based Fact Verification*. 2020. arXiv: 1909.02164 [cs.CL]. URL: <https://arxiv.org/abs/1909.02164>.
- [9] Zhiyu Chen et al. *FinQA: A Dataset of Numerical Reasoning over Financial Data*. 2022. arXiv: 2109.00122 [cs.CL]. URL: <https://arxiv.org/abs/2109.00122>.
- [10] Yew Ken Chia et al. *Contrastive Chain-of-Thought Prompting*. 2023. arXiv: 2311.09277 [cs.CL]. URL: <https://arxiv.org/abs/2311.09277>.
- [11] Primakov Chungkham et al. *Think Right, Not More: Test-Time Scaling for Numerical Claim Verification*. 2025. arXiv: 2509.22101 [cs.CL]. URL: <https://arxiv.org/abs/2509.22101>.
- [12] J. Cohen. "A Coefficient of Agreement for Nominal Scales". In: *Educational and Psychological Measurement* 20.1 (1960), p. 37.
- [13] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948 [cs.CL]. URL: <https://arxiv.org/abs/2501.12948>.
- [14] Shehzaad Dhuliawala et al. *Chain-of-Verification Reduces Hallucination in Large Language Models*. 2023. arXiv: 2309.11495 [cs.CL]. URL: <https://arxiv.org/abs/2309.11495>.
- [15] Jonathan St.B.T. Evans. "In two minds: dual-process accounts of reasoning". In: *Trends in Cognitive Sciences* 7.10 (2003), pp. 454–459. ISSN: 1364-6613. DOI: <https://doi.org/10.1016/j.tics.2003.08.012>. URL: <https://www.sciencedirect.com/science/article/pii/S1364661303002250>.
- [16] Jiahai Feng and Jacob Steinhardt. *How do Language Models Bind Entities in Context?* 2024. arXiv: 2310.17191 [cs.LG]. URL: <https://arxiv.org/abs/2310.17191>.
- [17] Yunfan Gao et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2312.10997 [cs.CL]. URL: <https://arxiv.org/abs/2312.10997>.
- [18] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: <https://arxiv.org/abs/2407.21783>.
- [19] Jiawei Gu et al. *A Survey on LLM-as-a-Judge*. 2025. arXiv: 2411.15594 [cs.CL]. URL: <https://arxiv.org/abs/2411.15594>.

- [20] Taneesh Gupta et al. *CARMO: Dynamic Criteria Generation for Context-Aware Reward Modelling*. 2025. arXiv: 2410.21545 [cs.CL]. URL: <https://arxiv.org/abs/2410.21545>.
- [21] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: <https://arxiv.org/abs/2106.09685>.
- [22] Lei Huang et al. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions". In: *ACM Transactions on Information Systems* 43.2 (Jan. 2025), pp. 1–55. ISSN: 1558-2868. DOI: 10.1145/3703155. URL: <http://dx.doi.org/10.1145/3703155>.
- [23] Ryo Kamoi et al. *Evaluating LLMs at Detecting Errors in LLM Responses*. 2024. arXiv: 2404.03602 [cs.CL]. URL: <https://arxiv.org/abs/2404.03602>.
- [24] Ryo Kamoi et al. "When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs". In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 1417–1440. DOI: 10.1162/tacl_a_00713. URL: <https://aclanthology.org/2024.tacl-1.78/>.
- [25] Muhammad Khalifa et al. *Process Reward Models That Think*. 2025. arXiv: 2504.16828 [cs.LG]. URL: <https://arxiv.org/abs/2504.16828>.
- [26] Tushar Khot et al. *Decomposed Prompting: A Modular Approach for Solving Complex Tasks*. 2023. arXiv: 2210.02406 [cs.CL]. URL: <https://arxiv.org/abs/2210.02406>.
- [27] Lukas Kirchdorfer et al. *Analytical Uncertainty-Based Loss Weighting in Multi-Task Learning*. 2024. arXiv: 2408.07985 [cs.LG]. URL: <https://arxiv.org/abs/2408.07985>.
- [28] Jinu Lee et al. *Evaluating Legal Reasoning Traces with Legal Issue Tree Rubrics*. 2025. arXiv: 2512.01020 [cs.AI]. URL: <https://arxiv.org/abs/2512.01020>.
- [29] Michael A. Lepori, Michael C. Mozer, and Asma Ghandeharioun. *Racing Thoughts: Explaining Contextualization Errors in Large Language Models*. 2025. arXiv: 2410.02102 [cs.CL]. URL: <https://arxiv.org/abs/2410.02102>.
- [30] Kin Kwan Leung et al. *Classifying and Addressing the Diversity of Errors in Retrieval-Augmented Generation Systems*. 2026. arXiv: 2510.13975 [cs.CL]. URL: <https://arxiv.org/abs/2510.13975>.
- [31] Haoyang Li et al. *Exposing Numeracy Gaps: A Benchmark to Evaluate Fundamental Numerical Abilities in Large Language Models*. 2025. arXiv: 2502.11075 [cs.CL]. URL: <https://arxiv.org/abs/2502.11075>.
- [32] Hunter Lightman et al. *Let's Verify Step by Step*. 2023. arXiv: 2305.20050 [cs.LG]. URL: <https://arxiv.org/abs/2305.20050>.
- [33] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: <https://arxiv.org/abs/1711.05101>.
- [34] Aman Madaan et al. *Self-Refine: Iterative Refinement with Self-Feedback*. 2023. arXiv: 2303.17651 [cs.CL]. URL: <https://arxiv.org/abs/2303.17651>.
- [35] Sourab Mangrulkar et al. *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods*. <https://github.com/huggingface/peft>. 2022.
- [36] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit." In: *ACL (System Demonstrations)*. The Association for Computer Linguistics, 2014, pp. 55–60. ISBN: 978-1-941643-00-6. URL: <http://dblp.uni-trier.de/db/conf/acl/acl2014-d.html#ManningSBFBM14>.
- [37] Swaroop Mishra et al. *NumGLUE: A Suite of Fundamental yet Challenging Mathematical Reasoning Tasks*. 2022. arXiv: 2204.05660 [cs.CL]. URL: <https://arxiv.org/abs/2204.05660>.
- [38] Niklas Muennighoff et al. *s1: Simple test-time scaling*. 2025. arXiv: 2501.19393 [cs.CL]. URL: <https://arxiv.org/abs/2501.19393>.
- [39] Kevin Nanekhan et al. *FlashCheck: Exploration of Efficient Evidence Retrieval for Fast Fact-Checking*. 2025. arXiv: 2502.05803 [cs.IR]. URL: <https://arxiv.org/abs/2502.05803>.

- [40] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG]. URL: <https://arxiv.org/abs/1912.01703>.
- [41] Python Software Foundation. *Python*. Version 3.10.9. Dec. 6, 2022. URL: <https://www.python.org/>.
- [42] Rafael Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. 2024. arXiv: 2305.18290 [cs.LG]. URL: <https://arxiv.org/abs/2305.18290>.
- [43] Valerie F. Reyna and Charles J. Brainerd. “Numeracy, gist, literal thinking and the value of nothing in decision making”. In: *Nature Reviews Psychology* (May 2023). Advance online publication, pp. 1–19. ISSN: 2731-0574. DOI: 10.1038/s44159-023-00188-7.
- [44] Rachneet Sachdeva et al. *Localizing and Mitigating Errors in Long-form Question Answering*. 2025. arXiv: 2407.11930 [cs.CL]. URL: <https://arxiv.org/abs/2407.11930>.
- [45] Taku Sakamoto, Saku Sugawara, and Akiko Aizawa. “Development of Numerical Error Detection Tasks to Analyze the Numerical Capabilities of Language Models”. In: *Proceedings of the 31st International Conference on Computational Linguistics*. Ed. by Owen Rambow et al. Abu Dhabi, UAE: Association for Computational Linguistics, Jan. 2025, pp. 9957–9976. URL: <https://aclanthology.org/2025.coling-main.666/>.
- [46] Parshin Shojaee et al. *The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity*. 2025. arXiv: 2506.06941 [cs.AI]. URL: <https://arxiv.org/abs/2506.06941>.
- [47] Charlie Snell et al. *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*. 2024. arXiv: 2408.03314 [cs.LG]. URL: <https://arxiv.org/abs/2408.03314>.
- [48] Peiyang Song, Pengrui Han, and Noah Goodman. “A Survey on Large Language Model Reasoning Failures”. In: *2nd AI for Math Workshop @ ICML 2025*. 2025. URL: <https://openreview.net/forum?id=hsgMn4KBFG>.
- [49] Pragya Srivastava et al. *Evaluating LLMs’ Mathematical Reasoning in Financial Document Question Answering*. 2025. arXiv: 2402.11194 [cs.CL]. URL: <https://arxiv.org/abs/2402.11194>.
- [50] Yang Sui et al. *Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models*. 2025. arXiv: 2503.16419 [cs.CL]. URL: <https://arxiv.org/abs/2503.16419>.
- [51] LM Studio Team. *LM Studio: Discover, Download and Run Local LLMs*. Desktop app for running large language models locally (Windows, macOS, Linux). 2023. URL: <https://lmstudio.ai/>.
- [52] Hieu Tran et al. *RARE: Retrieval-Augmented Reasoning Enhancement for Large Language Models*. 2025. arXiv: 2412.02830 [cs.CL]. URL: <https://arxiv.org/abs/2412.02830>.
- [53] Venkatesh V et al. *QuanTemp: A real-world open-domain benchmark for fact-checking numerical claims*. 2024. arXiv: 2403.17169 [cs.CL]. URL: <https://arxiv.org/abs/2403.17169>.
- [54] Venkatesh V., Deepali Prabhu, and Avishek Anand. *DEXTER: A Benchmark for open-domain Complex Question Answering using LLMs*. 2024. arXiv: 2406.17158 [cs.CL]. URL: <https://arxiv.org/abs/2406.17158>.
- [55] V Venkatesh, Deepali Prabhu, and Avishek Anand. *A Benchmark for Open-Domain Numerical Fact-Checking Enhanced by Claim Decomposition*. 2025. arXiv: 2510.22055 [cs.IR]. URL: <https://arxiv.org/abs/2510.22055>.
- [56] Wenhui Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. arXiv: 2002.10957 [cs.CL]. URL: <https://arxiv.org/abs/2002.10957>.
- [57] Xuezhi Wang et al. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. 2023. arXiv: 2203.11171 [cs.CL]. URL: <https://arxiv.org/abs/2203.11171>.
- [58] Yaoting Wang et al. *Multimodal Chain-of-Thought Reasoning: A Comprehensive Survey*. 2025. arXiv: 2503.12605 [cs.CV]. URL: <https://arxiv.org/abs/2503.12605>.
- [59] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: <https://arxiv.org/abs/2201.11903>.

- [60] Thomas Wolf et al. *HuggingFace’s Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL]. URL: <https://arxiv.org/abs/1910.03771>.
- [61] Ran Xu et al. *Alternating Reinforcement Learning for Rubric-Based Reward Modeling in Non-Verifiable LLM Post-Training*. 2026. arXiv: 2602.01511 [cs.CL]. URL: <https://arxiv.org/abs/2602.01511>.
- [62] Rongwu Xu et al. *Knowledge Conflicts for LLMs: A Survey*. 2024. arXiv: 2403.08319 [cs.CL]. URL: <https://arxiv.org/abs/2403.08319>.
- [63] An Yang et al. “Qwen2.5 Technical Report”. In: *arXiv preprint arXiv:2412.15115* (2024).
- [64] An Yang et al. *Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement*. 2024. arXiv: 2409.12122 [cs.CL]. URL: <https://arxiv.org/abs/2409.12122>.
- [65] Shunyu Yao et al. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. 2023. arXiv: 2305.10601 [cs.CL]. URL: <https://arxiv.org/abs/2305.10601>.
- [66] Seonghyeon Ye et al. *FLASK: Fine-grained Language Model Evaluation based on Alignment Skill Sets*. 2024. arXiv: 2307.10928 [cs.CL]. URL: <https://arxiv.org/abs/2307.10928>.
- [67] Jian Zhao et al. *GenPRM: Scaling Test-Time Compute of Process Reward Models via Generative Reasoning*. 2025. arXiv: 2504.00891 [cs.CL]. URL: <https://arxiv.org/abs/2504.00891>.
- [68] Yibo Zhao et al. *GRACE: Reinforcement Learning for Grounded Response and Abstention under Contextual Evidence*. 2026. arXiv: 2601.04525 [cs.CL]. URL: <https://arxiv.org/abs/2601.04525>.
- [69] Denny Zhou et al. *Least-to-Most Prompting Enables Complex Reasoning in Large Language Models*. 2023. arXiv: 2205.10625 [cs.AI]. URL: <https://arxiv.org/abs/2205.10625>.
- [70] Fengbin Zhu et al. “TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 3277–3287. DOI: 10.18653/v1/2021.acl-long.254. URL: <https://aclanthology.org/2021.acl-long.254>.
- [71] Andreas Ziegler and Maren Vens. “Generalized Estimating Equations”. In: *Handbook of Epidemiology*. Ed. by Wolfgang Ahrens and Iris Pigeot. New York, NY: Springer New York, 2014, pp. 1337–1376. ISBN: 978-0-387-09834-0. DOI: 10.1007/978-0-387-09834-0_45. URL: https://doi.org/10.1007/978-0-387-09834-0_45.



Appendix A

A.1. Human Qualitative Analysis Guidelines

For RQ1, human annotators were asked to inspect reasoning traces for three types of reasoning failures: numerical scoping errors, arithmetic errors, and looping errors. Each annotation instance contained the original claim or question, the evidence supplied to the model, the dataset label or answer, the model's reasoning trace, the model's final answer, and the model's justification. Annotators were asked to determine whether each error type appeared in the reasoning trace, and, when applicable, whether the error contributed to an incorrect final answer. Annotation guidelines were refined over time, resulting in the following formulation.

A.1.1. Numerical Scoping Errors

Numerical scoping errors were defined as cases where a numerical quantity was incorrectly bound to the wrong referent, timeframe, unit, comparison, or contextual meaning. In other words, the model used a number in a way that did not match what the input query, evidence, or output reasoning supported. Annotators were instructed to examine each statistic mentioned in the reasoning trace and classify it as one of three types:

- **Extractive statistic:** a quantity copied or paraphrased from the input claim, question, or evidence;
- **Extrapolative statistic:** a quantity or conclusion derived by combining, comparing, or manipulating other statistics;
- **Parametric statistic:** a quantity introduced from the model's background knowledge rather than from the provided input or reasoning.

For extractive statistics, annotators checked whether the quantity was grounded in the input and whether its referent, unit, scale, and timeframe were preserved. For extrapolative statistics, annotators checked whether the underlying quantities were combined under compatible scopes. For parametric statistics, annotators checked whether the model introduced an unsupported numerical fact and treated it as if it were grounded in the provided evidence.

Annotators were also asked to inspect the final answer and justification to determine whether any scoping error in the reasoning trace carried through to impact the final prediction. See Figure 1.2 and Table 3.1 for examples of numerical scoping errors.

A.1.2. Arithmetic Errors

Arithmetic errors were defined as incorrect numerical calculations performed over otherwise identifiable quantities. These included incorrect addition, subtraction, multiplication, division, percentage calculation, averaging, or comparison. Annotators marked an arithmetic error when the model selected the relevant numbers but produced an incorrect calculation result.

Arithmetic errors were treated separately from scoping errors. For example, using the wrong timeframe or referent for a number was considered a scoping error, while correctly identifying the relevant numbers but computing their ratio incorrectly was considered an arithmetic error. See Figure 5.1 for examples of arithmetic errors.

A.1.3. Looping Errors

Looping errors were defined as cases where the reasoning trace repeated, contradicted, or restarted its reasoning without making coherent progress towards a stable answer. Annotators marked a looping error when the model cycled between alternative answers, repeatedly reformulated the same point, contradicted itself across excessive reasoning steps, or failed to converge before producing a final answer.

Looping errors were only annotated when the repetition or instability affected the reasoning process itself, rather than merely reflecting a short clarification or correction. Annotators also recorded whether the looping behavior contributed to an invalid or incorrect final answer.

A.1.4. Annotation Procedure

Annotators first read the claim or question to understand the required numerical scope. They then reviewed the reasoning trace step by step, marking any scoping, arithmetic, or looping errors and briefly noting where the error occurred. Finally, they compared the reasoning trace with the model’s final answer and justification to determine whether the identified error contributed to the final-output incorrectness. A small trial set was completed before the main annotation task to resolve ambiguities and calibrate the annotation procedure.

A.2. Mathematical Formulae Required for Methodology

A.2.1. Cosine Similarity

Cosine similarity is a widely used metric for comparing two non-zero vectors according to the angle between them, rather than their absolute magnitude. Given two vectors x and y , cosine similarity is defined as:

$$\text{cos_sim}(x, y) = \frac{x \cdot y}{\|x\| \|y\|} \quad (\text{A.1})$$

where $x \cdot y$ denotes the dot product between the two vectors, while $\|x\|$ and $\|y\|$ denote their Euclidean norms.

Cosine similarity takes values in the range $[-1, 1]$, where:

- 1 indicates that the vectors point in the same direction, corresponding to maximum similarity;
- 0 indicates that the vectors are orthogonal, corresponding to no angular similarity;
- -1 indicates that the vectors point in exactly opposite directions.

In this work, cosine similarity is used to compare latent representations of tokens, layers, or classes within the embedding space to retrieve evidence, few-shot examples, or de-duplicate traces. This provides an interpretable measure of semantic similarity between learned representations.

A.2.2. F1 Score & Averaging Methods

The F1 score is a classification metric that combines precision and recall into a single value, commonly used in scenarios where class distributions are imbalanced. For a given class, let TP , FP , and FN denote the number of true positives, false positives, and false negatives, respectively. Precision, recall, and F1 are computed as:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (\text{A.2})$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (\text{A.3})$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (\text{A.4})$$

For multi-class classification, the per-class F1 score $F1_i$ is computed separately for each class i . The macro-averaged F1 score treats all classes equally and is defined as:

$$F1_{\text{macro}} = \frac{1}{C} \sum_{i=1}^C F1_i \quad (\text{A.5})$$

where C is the number of classes. The weighted-averaged F1 score instead weights each class by its support:

$$F1_{\text{weighted}} = \sum_{i=1}^C \frac{n_i}{N} F1_i \quad (\text{A.6})$$

where n_i is the number of true instances of class i and $N = \sum_{i=1}^C n_i$ is the total number of instances. Per-class F1 scores are reported to show performance for each individual class, while macro-F1 and weighted-F1 summarize performance across all classes.

These averaging strategies are essential for understanding classifier performance in imbalanced datasets and help provide greater insight over simple accuracy metrics.

A.2.3. ROC-AUC & Label Agreement

ROC-AUC evaluates how well a continuous score ranks positive examples above negative examples across all possible classification thresholds. In this work, it is used for scoping-error detection, where each trace has a reference binary label $s_i \in \{0, 1\}$ and a predicted scoping score \tilde{s}_i . ROC-AUC can be written as:

$$\text{ROC-AUC} = \frac{1}{N_+ N_-} \sum_{i:s_i=1} \sum_{j:s_j=0} \left[\mathbb{I}(\tilde{s}_i > \tilde{s}_j) + \frac{1}{2} \mathbb{I}(\tilde{s}_i = \tilde{s}_j) \right], \quad (\text{A.7})$$

where N_+ and N_- are the number of positive and negative examples, respectively. A value of 1 indicates perfect ranking, 0.5 corresponds to random ranking, and values below 0.5 indicate worse-than-random ranking.

Label agreement measures the percentage of examples for which the predicted binary label matches the reference label. Given the reference label s_i and predicted labels \hat{s}_i for N examples, label agreement is defined as:

$$\text{Agreement} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{s}_i = s_i), \quad (\text{A.8})$$

where the predicted label \hat{s}_i is derived from the predicted scoping score \tilde{s}_i by thresholding it with threshold $\lambda = 0.4$: $\hat{s}_i = \mathbb{I}(\tilde{s}_i \geq \lambda)$.

In this work, ROC-AUC is used to evaluate the ranking quality of continuous ORM scoping scores, while label agreement is used after thresholding these scores into binary scoping-error predictions.

A.2.4. Answer Entropy & Disagreement

Answer entropy measures how spread out the generated candidate answers are for a given query. Let \hat{A}_x denote the set of unique valid answers generated for a given fact-checking claim or question-answering question x , and let $p_x(\hat{a}_i)$ be the proportion of valid traces for query x that produce answer \hat{a}_i . The normalized answer entropy is defined as:

$$E_{\text{norm}}(x) = -\frac{1}{\log |\hat{A}_x|} \sum_{\hat{a}_i \in \hat{A}_x} p_x(\hat{a}_i) \log p_x(\hat{a}_i), \quad (\text{A.9})$$

for $|\hat{A}_x| > 1$. When only one unique valid answer is generated, entropy is set to 0. Normalization bounds the value between 0 and 1, where higher values indicate that candidate answers are more evenly distributed.

Answer disagreement measures the proportion of valid candidate answers that do not belong to the majority answer for a query. It is defined as:

$$D(x) = 1 - \max_{\hat{a}_i \in \hat{A}_x} p_x(\hat{a}_i). \quad (\text{A.10})$$

A disagreement value of 0 indicates that all valid traces produced the same answer, while larger values indicate greater variation across candidate traces. In this work, both metrics are averaged over all queries to summarize answer diversity for each evidence setting and value of K .

A.3. LLM Prompts

The structures of the numerical fact-checking and QA task prompts are shown in Figures A.1 and A.2, respectively. The structures of the straight LLM-as-a-Judge task prompts for fact-checking and QA are shown in Figures A.3 and A.4, respectively. The structures of the extractive scoping verification, non-extractive scoping verification, and importance weighting task prompts for fact-checking and QA can be seen in Figures A.5, A.6, and A.7, respectively. The structure of the outcome reward model prompts are shown in Figure A.8.

Fact-Checking Prompt Structure

System Message:
You are a fact-checking assistant tasked with evaluating claims based on provided evidence. Your task:

1. Carefully analyze the important details from both the claim and the evidence provided.
2. Reason through the evidence step-by-step, synthesizing relevant information to assess the overall veracity of the claim.
3. Finally, classify the claim's veracity into one of the following categories:
 - ****SUPPORTS****: The evidence fully supports the claim.
 - ****REFUTES****: The evidence directly contradicts the claim.
 - ****CONFLICTING****: The evidence contains contradictory elements or is inconclusive.

Response Format:

- [Label]: (SUPPORTS, REFUTES, or CONFLICTING)
- [Justification]: The reasoning steps used that led to your classification of the claim.

User Message:
Here are examples of fact-checking:

1. [Claim]: ...
[Evidences]: ...
[Label]: ...
2. [Claim]: ...
[Evidences]: ...
[Label]: ...

Similar to the given examples, fact check the following claim using the evidence. Pay additional attention to numerical spans in claims and evidence and fact check by thinking step by step and output the label by performing entailment. Classify the entire claim strictly into one of the following categories: SUPPORTS, REFUTES, or CONFLICTING.

[Claim]: ...
[Evidences]: ...

Give the reply in the following format:
[Label]: (SUPPORTS, REFUTES, or CONFLICTING)
[Justification]:

Figure A.1: Structure of the numerical fact-checking prompt.

Question-Answering Prompt Structure

System Message:
You are a question-answering assistant tasked with answering questions based on provided evidence.
Your task:

1. Carefully analyze and decompose the important details from the question.
2. Reason through the evidence in the table and text step-by-step, looking for information to help you answer the question.
3. Finally, output the final answer to the question in numerical or binary form without extra text.

Response Format:

- [Final Answer]: (YES, NO, or some number)
- [Justification]: The reasoning steps used that led to your final answer.

User Message:
Here are examples of question answering:

1. [Question]: ...
[Evidences]: ...
[Final Answer]: ...
2. [Question]: ...
[Evidences]: ...
[Final Answer]: ...

Similar to the given examples, answer the following questions using the table and text evidence. Give the step by step solution justification. Output the final answer as binary (YES or NO) or as a number without additional text.

[Question]: ...
[Evidences]: ...

Give the reply in the following format:
[Final Answer]: (YES, NO, or some number)
[Justification]:

Figure A.2: Structure of the numerical QA prompt.

Straight LLM-as-a-Judge Prompt Structure for Fact-Checking

System Message:

You are an evaluation model tasked with analyzing the reasoning traces produced by a fact-checking LLM. You will be given:

1. A claim
2. Evidence used for fact-checking
3. A reasoning trace produced by an LLM
4. The LLM's predicted label (SUPPORTS / REFUTES / CONFLICTING), if present
5. The LLM's justification, if present
6. The human-annotated ground-truth label (SUPPORTS / REFUTES / CONFLICTING)

Your task is to determine whether the reasoning trace contains any of the following failure types and, if so, whether they contributed to an incorrect final prediction. **IMPORTANT:** Reasoning traces can contain no numerical reasoning failures. Do not infer failures unless there is clear evidence in the reasoning trace.

FAILURE TYPES

1. Numerical scoping error
 - A numerical quantity is incorrectly bound to an entity, attribute, unit, population, or scope different from what is supported by the evidence.
 - Examples: mixing up two different "rates," attaching a number to the wrong country/year, confusing "revenue" vs "profit," swapping numerator/denominator, applying a statistic to the wrong population, mismatching units (% , dollars, people, acres), or using evidence about one entity to justify a claim about another.
2. Arithmetic errors:
 - The reasoning combines quantities incorrectly (wrong addition/subtraction/multiplication/division), miscalculates differences/percent changes, misapplies rounding, double-counts, or performs invalid aggregation.
3. Infinite loops / no final verdict:
 - The reasoning trace fails to terminate or does not produce a final label due to repetitive reasoning.
 - If a final label is present and the trace ends cleanly, mark NO. If there is no [Label] block, mark YES.

CAUSALITY RULE

For each failure type, you must decide whether it contributed to an incorrect final answer:

- If the predicted label matches the ground-truth label, any detected failure **MUST** be marked as not contributing to an incorrect answer.
- If the predicted label does not match the ground-truth label:
 - "yes" if the failure plausibly caused or materially contributed to choosing the wrong label.
 - "no" if the failure exists but is irrelevant to why the final label is wrong.
 - If you are unsure about causality, choose "unclear" and explain why.

YOUR OUTPUT FORMAT (STRICT)

Return a single JSON object with exactly these top-level keys:

```
{
  'verdict_correct': boolean,           // whether predicted label equals ground-truth label (true or false)
  'failures': {
    'numerical_scoping_error': {
      'exists': 'yes'|'no',
      'where': [ { 'span_quote': string, 'explanation': string } ],
      'led_to_incorrect_answer': 'yes'|'no'|'unclear'
    },
    'arithmetic_error': {
      'exists': 'yes'|'no',
      'where': [ { 'span_quote': string, 'explanation': string } ],
      'led_to_incorrect_answer': 'yes'|'no'|'unclear'
    },
    'infinite_loop': {
      'exists': "yes"|"no",
      'where': [ { 'span_quote': string, 'explanation': string } ],
      'led_to_incorrect_answer': 'yes'|'no'|'unclear'
    }
  },
  'brief_summary': string              // 1-3 sentences, focusing on the most important detected issue(s)
}
```

RULES

- If "exists" is "no", "where" MUST be an empty list [].
- "span_quote" should be a short direct quote (or two) from the reasoning trace or justification that localizes the failure.
- Keep explanations concrete: name the numbers, entities, units, and what was mismatched or miscalculated.
- Do not output anything except the JSON.

User Message:

Judge the following fact-checking output for numerical reasoning failures. Do NOT re-run the full fact-check from scratch unless needed. Detect and precisely localize scoping, arithmetic, and looping failures in the fact-checking LLM's output, only if present. Output only the JSON with no other text.

== INPUTS ==

[Claim] ...

[Evidence] ...

[Fact-checking LLM Output: Reasoning Trace, LLM Predicted Label, and LLM Justification] ...

[Ground Truth Label (Human)] ...

== END INPUTS ==

== YOUR OUTPUT ==

Figure A.3: Structure of the Straight LLM-as-a-Judge numerical fact-checking prompt.

Straight LLM-as-a-Judge Prompt Structure for Question-Answering

System Message:

You are an evaluation model tasked with analyzing the reasoning traces produced by a question-answering LLM.

You will be given:

1. A question
2. Evidence used for answering
3. A reasoning trace produced by an LLM
4. The LLM's predicted answer ([Final Answer]: YES / NO / some number), if present
5. The LLM's justification ([Justification]: some text), if present
6. The ground-truth answer (YES / NO / some number)

Your task is to determine whether the reasoning trace contains any of the following failure types and, if so, whether they contributed to an incorrect final prediction. IMPORTANT: Reasoning traces can contain no numerical reasoning failures. Do not infer failures unless there is clear evidence in the reasoning trace.

FAILURE TYPES

1. Numerical scoping error
 - A numerical quantity is incorrectly bound to an entity, attribute, unit, population, or scope different from what is supported by the evidence.
 - Examples: mixing up two different "rates," attaching a number to the wrong country/year, confusing "revenue" vs "profit," swapping numerator/denominator, applying a statistic to the wrong population, mismatching units (% , dollars, people, acres), or using evidence about one entity to justify a claim about another.
2. Arithmetic errors:
 - The reasoning combines quantities incorrectly (wrong addition/subtraction/multiplication/division), miscalculates differences/percent changes, misapplies rounding, double-counts, or performs invalid aggregation.
3. Infinite loops / no final verdict:
 - The reasoning trace fails to terminate cleanly or does not produce a final answer due to repetitive reasoning.
 - If a final answer is present and the trace ends cleanly with [Final Answer]/[Justification] blocks, mark NO. If there is no [Final Answer] block, mark YES. The [Ground Truth Answer] block does not count as a [Final Answer] block.

CAUSALITY RULE

For each failure type, you must decide whether it contributed to an incorrect final answer:

- If the predicted answer matches the ground-truth answer, any detected failure MUST be marked as not contributing to an incorrect answer.
- If the predicted answer does not match the ground-truth answer:
- "yes" if the failure plausibly caused or materially contributed to the wrong answer.
- "no" if the failure exists but is irrelevant to why the final answer is wrong.
- If you are unsure about causality, choose "unclear" and explain why.

YOUR OUTPUT FORMAT (STRICT)

Return a single JSON object with exactly these top-level keys:

```
{
  'verdict_correct': boolean,           // whether predicted answer equals ground-truth answer (true or false)
  'failures': {
    'numerical_scoping_error': {
      'exists': 'yes'|'no',
      'where': [ { 'span_quote': string, 'explanation': string } ],
      'led_to_incorrect_answer': 'yes'|'no'|'unclear'
    },
    'arithmetic_error': {
      'exists': 'yes'|'no',
      'where': [ { 'span_quote': string, 'explanation': string } ],
      'led_to_incorrect_answer': 'yes'|'no'|'unclear'
    },
    'infinite_loop': {
      'exists': 'yes'|'no',
      'where': [ { 'span_quote': string, 'explanation': string } ],
      'led_to_incorrect_answer': 'yes'|'no'|'unclear'
    }
  },
  'brief_summary': string              // 1-3 sentences, focusing on the most important detected issue(s)
}
```

RULES

- If "exists" is "no", "where" MUST be an empty list [].
- "span_quote" should be a short direct quote (or two) from the reasoning trace or justification that localizes the failure.
- Keep explanations concrete: name the numbers, entities, units, and what was mismatched or miscalculated.
- Do not output anything except the JSON.

User Message:

Judge the following question-answering output for numerical reasoning failures. Do NOT re-answer the question from scratch unless needed. Detect and precisely localize scoping, arithmetic, and looping failures in the question-answering LLM's output, only if present. Output only the JSON with no other text.

```
== INPUTS ==
[Question] ...
[Evidence] ...
[Question-Answering LLM Output: Reasoning Trace, LLM Predicted Answer, and LLM Justification] ...
[Ground Truth Answer] ...
== END INPUTS ==
== YOUR OUTPUT ==
```

Figure A.4: Structure of the Straight LLM-as-a-Judge numerical question-answering prompt.

Extractive Statistic Scoping Verification Prompt Structure

System Message:
 You are an extractive quantity checker.
 Your ONLY task: for each STATISTIC, verify that it was correctly interpreted in THINKING after being extracted from the CLAIM QUESTION or EVIDENCE.
 Authoritative source: only CLAIM QUESTION and EVIDENCE. THINKING is only used to see how the statistic was interpreted by a model, not as a source of truth. The CLAIM QUESTION and EVIDENCE conflicting does not change whether a statistic was correctly interpreted.

For each statistic:

Step 1: Locate in INPUTS

- Find the exact sentences in CLAIM QUESTION or EVIDENCE that contain the full quantity expression (number + unit/scale words).
- span_quote(s) MUST consist of whole sentence(s) copied verbatim from CLAIM QUESTION or EVIDENCE. Never quote from THINKING or STATISTICS. The quote(s) must be at least a whole sentence.
- metadata contains the statistic's unit, scale, referent, and timeframe based on the span_quote. If a piece of metadata is unclear, mark that field as "unknown".
- If no exact substring exists, set input_matches=[] and correct="false".

Step 2: Check entailment
 Compare the INPUT span(s) and metadata to how the statistic is used in THINKING. The INPUT span and metadata must strictly entail the THINKING usage, meaning that surrounding context (such as units, referents, and scope) must be the same between the INPUT span and THINKING usage.
 Mark correct="false" if between INPUT span(s)/metadata and THINKING:

- unit or scale of statistic differs (million vs. billion, %, etc.)
- referent of statistic differs
- timeframe of statistic differs
- population/scope of statistic differs
- surrounding context of the statistic in the INPUT does not clearly support the interpreted meaning of the statistic in THINKING
- the INPUT sentence is truncated, malformed, or ambiguous and does not clearly map to the statistic_text in THINKING

correct="true" if the INPUT span(s)/metadata fully supports the statistic_text and its use in THINKING with no added assumptions.
 correct="unknown" if the INPUT text is insufficient to determine entailment.

Output MUST be ONLY valid JSON with this schema and no other text:

```
{
  "results": [
    {
      "statistic_id": <integer>,
      "statistic_quantity": "<original statistic number>",
      "statistic_text": "<original statistic text>",
      "input_matches": [
        {
          "input_source": "claim|question|evidence",
          "span_quote": "<verbatim input sentences>",
          "metadata": {
            "unit": "<unit>",
            "scale": "<scale>",
            "referent": "<who/what the number pertains to>",
            "timeframe": "<date or year>"
          }
        }
      ],
      "correct": "true|false|unknown",
      "justification":
        "<brief explanation why input_matches span_quote
        makes the statistic correct/incorrect>"
    }
  ]
}
```

User Message:
 For each STATISTIC 0-20, check if CLAIM QUESTION/EVIDENCE was interpreted correctly in THINKING. span_quote MUST be COPIED WHOLE sentences exactly from CLAIM QUESTION or EVIDENCE. Return only the JSON.

CLAIM: QUESTION:
 <CLAIM> <QUESTION>
 ...
 </CLAIM> </QUESTION>

EVIDENCES:
 <EVIDENCES>
 ...
 </EVIDENCES>

THINKING:
 <THINKING>
 ...
 </THINKING>

STATISTICS:
 <STATISTICS>
 ...
 </STATISTICS>

Figure A.5: Structure of the extractive statistic scoping verification prompt used in the rubric-based LLM-as-a-Judge pipeline. Task-specific differences are highlighted in blue for fact-checking and red for numerical question answering.

Non-Extractive Statistic Scoping Verification Prompt Structure

System Message:
 You are a quantity checker.
 Your only tasks per STATISTIC are:

- classify as "extrapolative" (derived or computed from other quantities in THINKING) or "parametric" (introduced as an assumption or from memory, not derived from other numbers).
- assess correctness:
 - extrapolative: whether the derivation is valid and grounded in its inputs (no hidden assumptions like mixing scopes/units/timeframes).
 - parametric: whether the number is reasonable commonsense vs extremely specific or outlandish/ungrounded.

For each STATISTIC:
Step 1: Determine type:

- If the statistic is computed or derived from other quantities in THINKING (approximation, sum, difference, ratio, percent, average, unit conversion, scaling) set `quantity_type="extrapolative"` and list the exact quantities used as input in `"dependencies"`.
- Else set `quantity_type="parametric"` and `dependencies=[]`.

Step 2: Determine extrapolative statistic's correctness & parametric statistic's plausibility:

- If `quantity_type="extrapolative"`,
 - Identify exactly which other quantities it was derived from and list these quantities in `dependencies`.
 - `correct="true"` if the derivation is mathematically valid, the inputs are compatible (same units, scale, etc.) and the resulting statistic is representative of its inputs (no hidden assumptions, no unit/denominator mismatch, no scope/timeframe mixing, no unjustified averaging/extrapolating).
 - `correct="false"` if math is wrong OR if it combines incompatible inputs OR if the derivation is ungrounded (e.g., mixing incomparable quantities, wrong units, wrong denominator, cherry-picking, extrapolating beyond scope).
 - `correct="unknown"` if derivation is unclear.
- If `quantity_type="parametric"`,
 - `correct="true"` if the number is commonsense / generic / clearly stated as an assumption, and not a specific hard-to-know factual number.
 - `correct="false"` if it is a specific hard-to-know factual number, outlandish number, or a precise factual claim with no supporting evidence. For example, if the statistic is something that an average college student wouldn't know from memory, mark `"false"`.
 - `correct="unknown"` if plausibility cannot be judged.

IMPORTANT:

- Do NOT **decide whether the claim is true** answer the question. Use the **CLAIM QUESTION** only to understand what the THINKING is about.
- For extrapolative statistics, you MUST list the quantities it depends on. If you cannot identify dependencies, set `quantity_type="parametric"` or `"unknown"` (and explain).

Return ONLY valid JSON with this schema:

```
{
  "results": [
    {
      "statistic_id": <int>,
      "statistic_quantity": "<original>",
      "statistic_text": "<original>",
      "quantity_type": "extrapolative|"parametric|"unknown",
      "dependencies": [
        "<text describing statistic used as input to derive current statistic>",
        ...
      ],
      "correct": "true|"false|"unknown",
      "justification": "<brief: why type + why correct/incorrect>"
    }
  ]
}
```

User Message:
 For each STATISTIC (IDs 0 - 20), classify STATISTIC as extrapolative vs parametric, then evaluate correctness per the system rules. Return ONLY the JSON.

CLAIM QUESTION:
 <CLAIM> <QUESTION>
 ...
 </CLAIM> </QUESTION>

EVIDENCES:
 <EVIDENCES>
 ...
 </EVIDENCES>

STATISTICS:
 <STATISTICS>
 ...
 </STATISTICS>

THINKING:
 <THINKING>
 ...
 </THINKING>

Figure A.6: Structure of the non-extractive statistic scoping verification prompt used in the rubric-based LLM-as-a-Judge pipeline. The prompt classifies non-extractive statistics as extrapolative or parametric and verifies their correctness or plausibility. Task-specific differences are highlighted in blue for fact-checking and red for numerical question answering.

Statistic Importance Weight Judging Prompt Structure

System Message:
 You are a quantity importance evaluator.
 Your ONLY task: for each STATISTIC, assign an importance_weight such that all weights sum to exactly 1.0.
 Importance means: how much this STATISTIC causally influences the FINAL_ANSWER and JUSTIFICATION (not how salient it is).

How to score (counterfactual + usage-based):

- A statistic gets HIGH weight if removing/changing it would likely change the FINAL_ANSWER (e.g., REFUTES → SUPPORTS/NEI) or substantially change the justification.
- A statistic gets LOW weight if it is merely mentioned, background, or redundant (its removal would not change the verdict).

Bias control:

- Do NOT overweight statistics just because they appear in the CLAIM QUESTION.
- Prefer statistics that are actually used as evidence for the verdict (esp. those in JUSTIFICATION).
- If a claim-numberquestion-number is only restated and not used in a comparison/calculation, it must receive low weight.

IMPORTANT:

- Do NOT evaluate whether the statistic is correct.
- Do NOT fact-check the claim answer the QUESTION.
- Only judge a statistic's influence on the reasoning and FINAL_ANSWER.

Return ONLY valid JSON:

```
{
  "results": [
    {
      "statistic_id": <int>,
      "statistic_quantity": "<original>",
      "statistic_text": "<original>",
      "importance_weight": <decimal>,
      "justification": "<brief explanation why this statistic has this level of influence>"
    }
  ]
}
```

Constraints:

- All importance_weight values must sum to exactly 1.
- Each weight must be between 0 and 1.

User Message:
 For each STATISTIC (IDs 0-20), rate how influential it is to the THINKING, FINAL_ANSWER, and JUSTIFICATION. All weights must sum to 1. You must assign a weight to EVERY STATISTIC. Return ONLY the JSON.

CLAIM: QUESTION:
 <CLAIM> <QUESTION>
 ...
 </CLAIM> </QUESTION>

EVIDENCES:
 <EVIDENCES>
 ...
 </EVIDENCES>

THINKING:
 <THINKING>
 ...
 </THINKING>

FINAL_ANSWER:
 <FINAL_ANSWER>
 ...
 </FINAL_ANSWER>

JUSTIFICATION:
 <JUSTIFICATION>
 ...
 </JUSTIFICATION>

STATISTICS:
 <STATISTICS>
 ...
 </STATISTICS>

Figure A.7: Structure of the statistic importance weight judging prompt used in the rubric-based LLM-as-a-Judge pipeline. The prompt assigns normalized statistic importance weights based on each statistic's causal influence on the final answer and justification. Task-specific differences are highlighted in blue for fact-checking and red for numerical question answering.

Reward Model Prompt Templates

Fact-checking Format:
 Claim: <CLAIM>
 Verdict: <VERDICT or UNKNOWN>
 Justification: <JUSTIFICATION or UNKNOWN>
 Thinking: <THINKING or UNKNOWN>

Numerical Question Answering Format:
 Question: <QUESTION>
 Final Answer: <FINAL ANSWER or UNKNOWN>
 Justification: <JUSTIFICATION or UNKNOWN>
 Thinking: <THINKING or UNKNOWN>

Figure A.8: Structure of the outcome reward model prompts for fact-checking (blue) and numerical question answering (red).

B

Appendix B

B.1. Training of Learned ORMs for Test-Time Scaling Experiments

Three variants of outcome reward models (ORMs) were trained in the process of this work: a general quality ORM based on correctness signals with ground truth, a scoping error ORM based on scoping error presence signals from the hybrid LLM-as-a-Judge pipeline, and a multi-task ORM that combines the aforementioned signals. The learning curves of each of these ORMs are shown in Figure B.1.

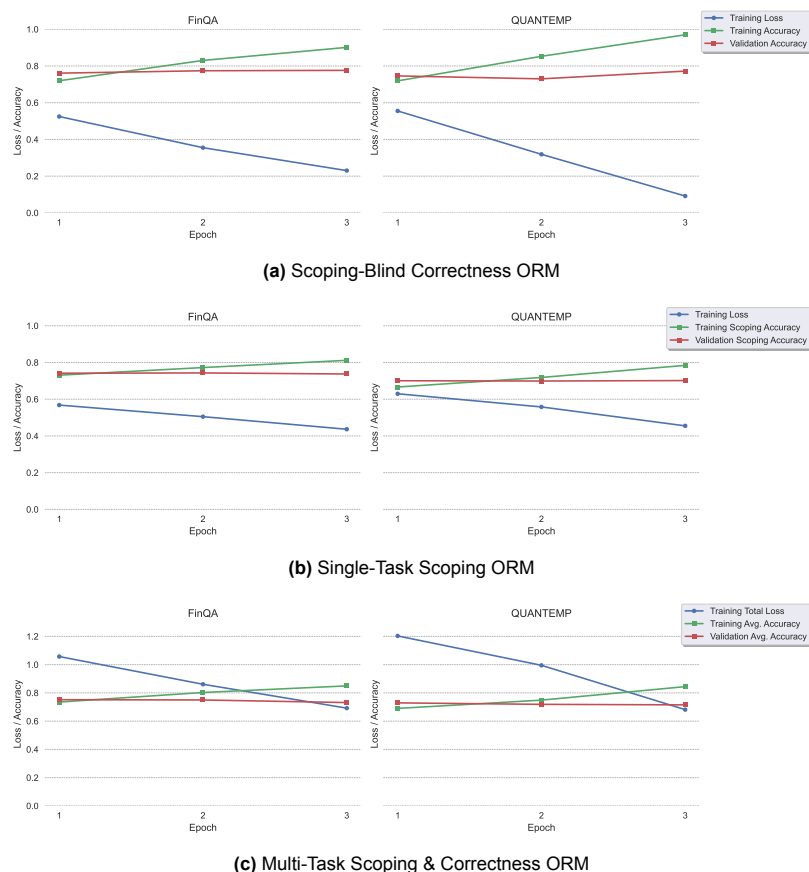


Figure B.1: Training learning curves for the learned outcome reward models (ORMs) used in test-time scaling experiments. The figures show training and validation accuracies and training loss for the scoping-blind correctness ORM, single-task scoping ORM, and multi-task scoping & correctness ORM. For the multi-task ORM, training loss is the total of both the correctness and scoping losses and accuracies are the unweighted average of the correctness and scoping accuracies.

None of the ORMs required more than two epochs to train with the early stopping patience of two. The task of learning correctness was easier than learning scoping error detection, as seen by the higher training and validation accuracies at higher epochs. None of the validation accuracies improved substantially over time, suggesting that the models reached their useful validation performance early in training. In contrast, the training loss and accuracy continued to improve, particularly for the correctness ORM, indicating overfitting after the first epochs. This supports the use of early stopping and suggests that additional training would be unlikely to improve downstream verifier quality.

B.2. Scoping Failure Detection Examples

Tables B.1, B.2, B.3, and B.4 show examples of the Hybrid LLM-as-a-Judge pipeline detecting scoping failures in LLM reasoning traces. Common themes are discussed in the “Explanation” column.

Table B.1: Example of numerical scoping failure detection by the Hybrid LLM-as-a-Judge Pipeline on a `FinQA` Whole-Document LLM reasoning trace. Mistakes in scoping failure detection are in red. Statistics are colored by type.

Inputs	Reasoning Trace	Statistic-Level Outputs	Trace-Level Label	Explanation
<p>Question: what portion of total purchase price is related to stock awards?</p> <p>Evidences: 1. "Text Before Table: - "chairman and a director of the board of fis as well as the chairman of the board of lps ." - "effective march 1 , 2010 , mr ." - "kennedy and the company mutually agreed that he would no longer serve as an executive officer and director of the company and its subsidiaries ." - "the revenue and expense items with lps are , therefore , summarized above as related party activity through march 1 , 2010 ."</p> <p>...10 more points</p> <p>- "metavante also added significant scale to treasury and cash management offerings and provided an entry into the health-care and government payments markets ." - "pursuant to the agreement and plan of merger dated as of march 31 , 2009 , metavante became a wholly-owned subsidiary of fis ." - "each issued and outstanding share of metavante common stock , par value \$ 0.01 per share , was converted into 1.35 shares of fis common stock ." - "in addition , outstanding metavante stock options and other stock-based awards were converted into comparable fis stock options and other stock-based awards at the same conversion ratio ." - "the total purchase price was as follows (in millions) : "</p> <p>2. "Table: value of metavante common stock\$ 4066.4 value of metavante stock awards 121.4 total purchase price\$ 4187.8"</p> <p>3. "Text After Table: - "we recorded a preliminary allocation of the purchase price to metavante tangible and identifiable intangible assets acquired and liabilities assumed based on their estimated fair values as of october 1 , 2009 ." - "goodwill was fidelity national information services , inc ." - "and subsidiaries notes to consolidated financial statements 2014 (continued) %>transmsg*** transmitting job : g26369 pcn : 064000000 ***%>pcmsgj64 [00007]yes[no]03/28/2011 17:32[0]page is valid , no graphics - color : n ."</p>	<p><think> Alright, so I need to figure out what portion of the total purchase price is related to stock awards. Let me start by looking at the provided evidence.</p> <p>First, I see that there's a table with three columns: net tangible assets, in-process R&D, developed technology and know-how, customer relationship, trade name, order backlog, deferred income taxes, goodwill, and estimated purchase price. The values are in millions. The total purchase price is \$4187.8 million.</p> <p>Looking at the table, the value of metavante stock awards is listed as \$121.4 million. This is under the "value of metavante stock awards" column.</p> <p>The question is asking for the portion of the total purchase price that is related to stock awards. So, I need to calculate what percentage \$121.4 million is of the total \$4187.8 million.</p> <p>To do this, I'll divide 121.4 by 4187.8. Let me do that calculation:</p> $121.4 \div 4187.8 \approx 0.029$ <p>To express this as a percentage, I multiply by 100, which gives approximately 2.9%.</p> <p>Looking at the table, the value of stock awards is a separate line item, so it's a direct part of the total purchase price. There's no mention of any adjustments or allocations beyond what's in the table, so I don't need to consider any other factors.</p> <p>Therefore, the portion of the total purchase price related to stock awards is about 2.9%, which I can round to 3% for simplicity.</think></p> <p>Final Answer: 3%</p> <p>Justification: The value of metavante stock awards is \$121.4 million, and the total purchase price is \$4187.8 million. Dividing 121.4 by 4187.8 gives approximately 2.9%, which rounds to 3%.</p>	<p>"three columns":</p> <ul style="list-style-type: none"> - Type: parametric - Scope-Valid: false (<i>The number is specific and not a reasonable assumption from the table which shows three columns but does not justify this as a meaningful statistic.</i>) - Importance Weight 0.00 (<i>This statistic is merely descriptive and does not influence the calculation or the final answer.</i>) <p>"\$ 4187.8 million":</p> <ul style="list-style-type: none"> - Type: parametric - Scope-Valid: true (<i>The number is directly stated in the table and is a reasonable assumption.</i>) - Importance Weight 0.40 (<i>This is the total purchase price, which is crucial for calculating the portion related to stock awards.</i>) <p>"4187.8":</p> <ul style="list-style-type: none"> - Type: parametric - Scope-Valid: true (<i>The number is directly stated in the table and is a reasonable assumption.</i>) - Importance Weight 0.00 (<i>This is redundant with statistic 2 and does not add new information.</i>) <p>"\$ 121.4 million":</p> <ul style="list-style-type: none"> - Type: parametric - Scope-Valid: true (<i>The number is directly stated in the table and is a reasonable assumption.</i>) - Importance Weight 0.40 (<i>This is the value of metavante stock awards, which is essential for calculating the portion of the total purchase price.</i>) <p>"121.4":</p> <ul style="list-style-type: none"> - Type: extractive - Scope-Valid: true (<i>The statistic is correctly interpreted as the value of metavante stock awards in millions, which is used to calculate the portion of the total purchase price.</i>) - Importance Weight 0.00 (<i>This is redundant with statistic 4 and does not add new information.</i>) <p>"100":</p> <ul style="list-style-type: none"> - Type: parametric - Scope-Valid: false (<i>The number is specific and not a reasonable assumption from the context.</i>) - Importance Weight 0.00 (<i>This is a constant used in percentage calculation and does not influence the final answer.</i>) <p>"about 2.9%":</p> <ul style="list-style-type: none"> - Type: extrapolative - Scope-Valid: true (<i>The derivation is valid and grounded in the inputs from the table.</i>) - Importance Weight 0.10 (<i>This is an intermediate result in the calculation, but not the final answer.</i>) <p>"3%":</p> <ul style="list-style-type: none"> - Type: extrapolative - Scope-Valid: true (<i>The derivation is valid and grounded in the inputs from the table.</i>) - Importance Weight 0.10 (<i>This is the final answer, rounded from the intermediate result.</i>) 	No	<p>This example shows a typical theme in the hybrid pipeline that the statistic de-duplication and matching for determining extractive statistics can often be too harsh, especially when processing tabular extractive statistics. "\$ 4187 million", "4187.8", "\$ 121.4 million", and "121.4" should all be extractive statistics, as they are various versions of input statistics "\$ 4187.8" and "121.4", however, due to the limited context found in the tabular evidence, only one of these statistics actually gets identified as extractive. "\$ 4187.8 million" and "\$ 121.4 million" are missed because of the additional "million" scaling which the table does not make clear unless the "(in millions)" gets included from the "Text Before Table" section. "4187.8" is not marked as extractive because in the input, a "\$" is included in front. Resulting from this, the statistics are not able to be de-duplicated and half get marked as unimportant by the weighting step. Although not an issue in this particular example because all these statistics were marked as scope-valid regardless, in other examples this could cause an incorrect verification of an extractive statistic being treated as parametric. Perhaps, in the future, an additional off-line input table statistic translation step can aid the extractive statistic verification.</p> <p>Next, there are some small less important mistakes, the first of which concerns an incorrect explanation for why "three columns" is mis-scoped. The table actually only has three columns and two rows, not two columns and three rows, meaning the judge misunderstood the table structure. The second less critical mistake in this example is marking "100" as mis-scoped. "100" is commonsense for a percentage calculation, the judge was too harsh with judging this parametric statistic.</p> <p>Finally, the whole trace gets marked as scope-valid, which is debatable depending on how important an annotator would consider the "three columns" mistake. This example also shows how scoping labels can be nuanced depending on the associated importance of the mistake.</p>

Table B.2: Another example of numerical scoping failure detection by the Hybrid LLM-as-a-Judge Pipeline on a `FinQA Whole-Document` LLM reasoning trace. Mistakes in scoping failure detection are in red. Statistics are colored by type.

Inputs	Reasoning Trace	Statistic-Level Outputs	Trace-Level Label	Explanation
<p>Question: what is the ratio of the total flight attendants to total maintenance personnel?</p> <p>Evidences: 1. "Text Before Table: - "table of contents configuration , amenities provided to passengers , loyalty programs , the automation of travel agent reservation systems , on-board products , markets served and other services ." - "we compete with both major network airlines and low-cost carriers throughout our network ." - "international in addition to our extensive domestic service , we provide international service to canada , central and south america , asia , europe , australia and new zealand ."</p> <p>...16 more points</p> <p>- "a peb examines the parties 2019 positions and recommends a solution ." - "the peb process lasts for 30 days and (if no resolution is reached) is followed by another 201ccooling off 201d period of 30 days ." - "at the end of a 201ccooling off 201d period (unless an agreement is reached , a peb is established or action is taken by congress) , the labor organization may exercise 201cself-help , 201d such as a strike , and the airline may resort to its own 201cself-help , 201d including the imposition of any or all of its proposed amendments to the cba and the hiring of new employees to replace any striking workers ." - "the table below presents our approximate number of active full-time equivalent employees as of december 31 , 2016 ." - "mainline operations wholly-owned regional carriers total ."</p> <p>2. "Table: mainline operations wholly-owned regional carriers total pilots and flight crew training instructors 13400 3400 16800 flight attendants 24700 2200 26900 maintenance personnel 14900 2000 16900 fleet service personnel 16600 3500 20100 passenger service personnel 15900 7100 23000 administrative and other 16000 2600 18600 total 101500 20800 122300"</p> <p>3. "Text After Table:"</p>	<p><think> Alright, I need to figure out the ratio of the total flight attendants to the total maintenance personnel using the provided table. Let me start by locating the relevant numbers in the table. Looking at the table, under the "mainline operations" column, I see the number of flight attendants is 24,700. For maintenance personnel, it's 14,900.</p> <p>Now, I need to calculate the ratio of flight attendants to maintenance personnel. That means I'll divide the number of flight attendants by the number of maintenance personnel. So, 24,700 divided by 14,900. Let me do that calculation.</p> <p>$24,700 \div 14,900 \approx 1.658$.</p> <p>Looking at the previous examples, the answers were given as numbers without additional text, so I'll present this ratio as is. Therefore, the ratio is approximately 1.66. But since the previous answers were given to two decimal places, I'll round it to two decimal places as well, making it 1.66. However, checking the examples, sometimes they used exact fractions or simplified numbers. Let me see if $24700/14900$ simplifies.</p> <p>Dividing numerator and denominator by 100 gives $247/149$. Let me check if this can be simplified further. 247 divided by 13 is 19, and 149 is a prime number. So, $247/149$ is the simplified fraction, which is approximately 1.66.</p> <p>Since the question asks for the ratio, I can present it as a decimal or a fraction. But in the examples, they used decimals. So, I'll go with 1.66 as the final answer.</think></p> <p>Final Answer: 1.66%</p> <p>Justification: The table shows 24,700 flight attendants and 14,900 maintenance personnel. Dividing 24,700 by 14,900 gives approximately 1.66.</p>	<p>"24,700": - Type: extractive - Scope-Valid: true (The statistic is correctly interpreted as the number of flight attendants from the table.) - Importance Weight 0.39 (This number is crucial as it represents the total number of flight attendants, which is a key component in calculating the ratio.)</p> <p>"14,900": - Type: extractive - Scope-Valid: true (The statistic is correctly interpreted as the number of maintenance personnel from the table.) - Importance Weight 0.39 (This number is crucial as it represents the total number of maintenance personnel, which is a key component in calculating the ratio.)</p> <p>"1.658": - Type: extrapolative - Scope-Valid: true ((The statistic is derived correctly from the given numbers in the table.) - Importance Weight 0.10 (This is the exact ratio calculated from the flight attendants and maintenance personnel numbers, but it is not the final answer presented.)</p> <p>"1.66": - Type: extrapolative - Scope-Valid: true (The statistic is derived correctly from the given numbers in the table.) - Importance Weight 0.05 (This is the final answer presented after rounding the exact ratio to two decimal places.)</p> <p>"two decimal places": - Type: parametric - Scope-Valid: true (The number of decimal places is a reasonable assumption based on the examples provided.) - Importance Weight 0.02 (This specifies the precision of the final answer but does not influence the calculation.)</p> <p>"100": - Type: parametric - Scope-Valid: false ((The number 100 is not derived from any given data and is out of context.) - Importance Weight 0.01 (This is used to simplify the numbers but does not influence the final ratio.)</p> <p>"247/149": - Type: extrapolative - Scope-Valid: true ((The statistic is derived correctly from the given numbers in the table.) - Importance Weight 0.01 (This is the simplified fraction of the ratio but does not influence the final answer.)</p> <p>"247": - Type: extrapolative - Scope-Valid: true (The statistic is derived correctly from the given number in the table.) - Importance Weight 0.01 (This is part of the simplified fraction but does not influence the final answer.)</p> <p>"13": - Type: parametric - Scope-Valid: false (The number 13 is not derived from any given data and is out of context.) - Importance Weight 0.01 (This is used in simplifying the fraction but does not influence the final answer.)</p> <p>"19": - Type: parametric - Scope-Valid: false (The number 19 is not derived from any given data and is out of context.) - Importance Weight 0.01 (This is part of the simplified fraction but does not influence the final answer.)</p> <p>"149": - Type: extrapolative - Scope-Valid: true ((The statistic is derived correctly from the given number in the table.) - Importance Weight 0.01 (This is part of the simplified fraction but does not influence the final answer.)</p>	No	<p>This example shows a typical missed scoping error and a common theme: the hybrid judge occasionally falls into the same contextualization mistake traps that the generation model does. "24,700" and "14,900" are indeed the number of flight attendants and maintenance personnel from the table, but they are specifically from the "mainline operations" column, not the "total" column. The hybrid judge misses this extractive scoping error and thus the trace gets marked as scope-valid when it should certainly be mis-scoped. Future work should analyze such false negative examples to determine ways to improve scoping failure detection.</p> <p>Furthermore, there are some less important mistakes: "100" again gets marked as mis-scoped even though it's a commonsense statistic used in simplifying fractions. The judge can sometimes be too harsh with judging certain statistics. Finally, "19" gets marked as parametric even though it is the result of some calculation and should be classified as extrapolative, leading to it being marked as mis-scoped even though it is mostly scope-valid. Regardless, these small mistakes do not influence the final trace-level label as they are justly marked as unimportant.</p>

Table B.3: Example of numerical scoping failure detection by the Hybrid LLM-as-a-Judge Pipeline on a QUANTEMP *Re-Ranked* LLM reasoning trace. Mistakes in scoping failure detection are in red. Statistics are colored by type.

Inputs	Reasoning Trace	Statistic-Level Outputs	Trace-Level Label	Explanation
<p>Claim: Company affiliated with Matt Gaetz had \$482.321 in PPP loans forgiven</p> <p>Evidences: 1. Fact check: did matt gaetz have \$482k in ppp loans forgiven even though he doesn't own a business? report: watchdog finds over \$5 billion in potential covid ..."</p> <p>2. "the white house tweeted that rep. matt gaetz (-fla.) had forgiven ppp loans of \$482,321 though that was in response to the lawmaker's"</p> <p>3. "matt gaetz had \$987237 and \$482321, respectively, in ppp loans forgiven many of the lawmakers pushing for the bill owned businesses that"</p>	<p><think> Alright, let's tackle this fact-checking task. The claim is that "Company affiliated with Matt Gaetz had \$482.321 in PPP loans forgiven." I need to evaluate this based on the provided evidence. First, I'll look at the evidence given. The first evidence mentions that Matt Gaetz had over \$5 billion in PPP loans forgiven, which is a significant amount. The second evidence directly states that he had \$482.321 in PPP loans forgiven. The third evidence talks about multiple amounts, including \$482,321, which suggests that this figure is accurate. Now, the claim is about a company affiliated with Matt Gaetz having exactly \$482.321 in PPP loans forgiven. The evidence supports that this specific amount was indeed forgiven. However, the claim doesn't specify whether this was for the company or for Matt Gaetz personally. The evidence points to the company, but it's not explicitly stated in the claim.</p> <p>Considering the numerical data, the evidence aligns perfectly with the claim's figure. The claim's veracity is supported because the exact amount mentioned in the evidence matches the claim. The fact that it's a company affiliated with Gaetz is also consistent with the evidence provided. I don't see any contradictions in the evidence. All the relevant numbers point to the claim being accurate. Therefore, the evidence fully supports the claim.</think></p> <p>Final Verdict: SUPPORTS</p> <p>Justification: The evidence directly states that Matt Gaetz had \$482,321 in PPP loans forgiven, which aligns with the claim about the company affiliated with him. The numerical data matches, supporting the claim's accuracy.</p>	<p>"\$ 482,321":</p> <ul style="list-style-type: none"> - Type: extractive - Scope-Valid: false (<i>The evidence mentions that Matt Gaetz had \$482,321 in PPP loans forgiven, but the claim specifies a company affiliated with Matt Gaetz. The referent differs between the claim and evidence.</i>) - Importance Weight 0.80 (<i>This statistic is crucial as it directly matches the amount mentioned in the claim and is explicitly stated in the evidence, supporting the claim's accuracy.</i>) <p>"over \$ 5 billion":</p> <ul style="list-style-type: none"> - Type: extractive - Scope-Valid: false (<i>The evidence mentions over \$5 billion in potential PPP loans, but this amount is not directly related to Matt Gaetz's specific PPP loan amount of \$482,321. The referent and timeframe differ.</i>) - Importance Weight 0.20 (<i>This statistic provides context but does not directly support or refute the specific claim amount. It is less influential in determining the final answer.</i>) 	Yes	<p>This example shows two typical issues with QUANTEMP scoping error detection. First, the judge finds it very difficult to stop fact-checking the claim and rather focus on verifying numerical contexts. The two tasks are, of course, interrelated but the judge often focuses too much on comparing the claim and the evidence rather than thinking about the reasoning. In this example, although the statistics do receive the correct mis-scoped label, the explanations show that the judge still over-focuses on the discrepancy between the claim and evidence rather analyzing the intricacies in the thinking.</p> <p>Another typical issue with QUANTEMP scoping error detection is the relatively low number of statistics. There are only two statistics in this analysis, which are mentioned frequently in the reasoning and evidence. Although the judge was able to correctly identify each quantity as mis-scoped it does so for the wrong reasons, often making the same mistakes that the original generation model made: mis-scoping the "\$ 5 billion" statistic as about PPP loans instead of something related to "covid". The low number of statistics in QUANTEMP means that the judge has fewer perspectives from which to analyze the reasoning and fewer chances to catch detrimental errors. Furthermore, in low-statistic traces, sometimes there are no truly important statistics, making the importance weighting normalization somewhat artificial: even weakly relevant quantities must receive non-zero relative weight because the weights are constrained to sum to one. Perhaps, a relative importance measure could be proposed to account for cases where there truly are no relevant statistics.</p>

Table B.4: Another example of numerical scoping failure detection by the Hybrid LLM-as-a-Judge Pipeline on a QUANTEMP *Re-Ranked* LLM reasoning trace. Mistakes in scoping failure detection are in red. Statistics are colored by type.

Inputs	Reasoning Trace	Statistic-Level Outputs	Trace-Level Label	Explanation
<p>Claim: Clip of French president slapped at meet-and-greet was filmed in November 2022</p> <p>Evidences: 1. "... when it is apparent that an interaction between a president and the press is clearly a press conference, on June 16, 2021, for example, president joe biden's ..."</p> <p>2. "Jun 8, 2021 french president emmanuel macron appears to get slapped in video. two men were immediately detained by police, a spokesman for the interior ..."</p> <p>3. "false. a clip of french president macron being slapped by a member of the public was not filmed in november 2022 and instead, shows a seemingly newly published angle of a June 2021 incident where macron was struck by a member of the public in tain-l'hermitage, auvergne-rhone-alpes. a clip of french president emmanuel macron being slapped across the face at a meet-and-greet with members of the public was filmed in June 2021, but users shared the clip in november 2022 with false claims that the president had been slapped again. the clip was not filmed recently, and instead, shows an angle filmed from the opposite side of a guard railing that separated the french president from members of the public on June 8, 2021."</p>	<p><think> Alright, let's tackle this fact-checking task step by step. The claim is that a clip of French President Emmanuel Macron being slapped at a meet-and-greet was filmed in November 2022. I need to evaluate this claim using the provided evidence. First, I'll look at the evidence given. The first evidence mentions that on June 16, 2021, President Joe Biden had a press interaction, but that's not relevant here. The second evidence is about June 8, 2021, where President Macron was slapped by two men, who were detained by police. The third evidence is a statement that refutes the claim, explaining that the clip was filmed in June 2021, not November 2022. It also mentions that the clip was shared in November 2022 with false claims, but the original filming was in June 2021.</p> <p>So, the key points are: - The incident happened in June 2021. - The clip was filmed then, not in November 2022. - The evidence explicitly states that the clip was not filmed in November 2022 but was shared later with false claims.</p> <p>Putting this together, the claim that the clip was filmed in November 2022 is false because the actual filming took place in June 2021. The evidence directly contradicts the claim by providing the correct year and explaining the misinformation spread afterward. Therefore, the claim is false, and the evidence refutes it.</think></p> <p>Final Verdict: REFUTES</p> <p>Justification: The evidence clearly states that the clip was filmed in June 2021, not November 2022. It also mentions that the clip was shared in November 2022 with false claims, but the original filming occurred in 2021. This directly contradicts the claim, thus refuting it.</p>	<p>"November 2022":</p> <ul style="list-style-type: none"> - Type: extractive - Scope-Valid: true (<i>The evidence clearly states that the clip was shared in November 2022 with false claims, which matches the statistic_text.</i>) - Importance Weight 0.30 (<i>This statistic is crucial as it directly contradicts the claim's date and is central to the refutation.</i>) <p>"June 16, 2021":</p> <ul style="list-style-type: none"> - Type: extractive - Scope-Valid: true (<i>The evidence mentions June 16, 2021, as a date when Biden had a press interaction, which matches the statistic_text.</i>) - Importance Weight 0.05 (<i>This date is mentioned but is not relevant to the main incident and does not influence the final answer.</i>) <p>"June 8, 2021":</p> <ul style="list-style-type: none"> - Type: extractive - Scope-Valid: true (<i>The evidence mentions June 8, 2021, as the date when Macron was slapped, which matches the statistic_text.</i>) - Importance Weight 0.40 (<i>This is the actual date of the incident and is critical to refuting the claim's incorrect date.</i>) <p>"two men":</p> <ul style="list-style-type: none"> - Type: extractive - Scope-Valid: true (<i>The evidence mentions that two men were detained by police, which matches the statistic_text.</i>) - Importance Weight 0.05 (<i>This detail is mentioned but does not influence the final answer or the main reasoning.</i>) <p>"June 2021":</p> <ul style="list-style-type: none"> - Type: extractive - Scope-Valid: true (<i>The evidence states that the clip was filmed in June 2021, which matches the statistic_text.</i>) - Importance Weight 0.20 (<i>This statistic is important as it clarifies the correct year of the incident and is used to refute the claim.</i>) 	No	<p>This example shows a typical low-difficulty scope-valid trace. All statistics were extracted correctly, classified correctly, and verified correctly. What is interesting to note is that the QUANTEMP <i>Re-Ranked</i> evidence actually contained evidence from a gold fact-checker website, meaning that there indeed is occasional temporal and gold evidence leakage as mentioned in Venkatesh, Prabhu, and Anand [55].</p>

B.3. Scoping-Aware Mitigation Examples

Tables B.5 and B.6 show examples of how the scoping-aware mitigation variants choose candidate reasoning traces in test-time scaling. Common themes are discussed in the “Explanation” column.

Table B.5: Example of LLM candidate traces chosen by the various test-time scaling variants on a QUANTEMP claim with *Re-Ranked* evidence. The Correctness Outcome Reward Model Best-of-N (C-ORM: BoN) variant chooses a trace based on a quality score; the Hybrid Pipeline Scoping Filtering + C-ORM BoN (HP: SF + C-ORM: BoN) variant chooses a trace based on first filtering for non-scoping failure traces and then choosing based on a quality score; the Scoping-ORM Scoping Filtering + C-ORM BoN (S-ORM: SF + C-ORM: BoN) variant filters traces based on a distilled scoping error signal, then chooses based on the C-ORM quality score; the Multi-Task ORM Scoping Filtering + BoN (MT-ORM: SF + BoN) variant filters traces based on a distilled scoping error signal, then chooses based on the MT-ORM’s quality score. Finally, the Multi-Task ORM Scoping & Correctness Best-of-N (MT-ORM: BoN) chooses a trace based on a combined scoping failure and correctness score. Scoping failures are highlighted in red. Verdicts are colored by correctness.

Inputs	Reasoning Traces				Explanation
	C-ORM: BoN & MT-ORM: BoN	HP: SF + C-ORM: BoN	S-ORM: SF + C-ORM: BoN	MT-ORM: SP + BoN	
<p>Claim: One in six Texans don't have health care. We're the most uninsured state in the U.S.</p> <p>Evidences: 1. "the census data shows that the five states with the highest uninsured rates did not expand its Medicaid eligibility. texas leads that category followed by oklahoma, georgia, wyoming and florida. the census data, released this week, showed that 18% of texans were uninsured in 2021. it was a very different story in massachusetts, where 2 1/2% of the residents were uninsured."</p> <p>2. "texas is the state with both the largest number and percentage of uninsured residents in the united states. texans make up 9% of the u.s. population, but 19% of the countrys uninsured population. nearly 1 in 4 working-age texans 19-64 is uninsured, making up the biggest share of texas uninsured, with younger adults at the highest likelihood of being uninsured. accept billions in federal funds to provide medicaid to working but poor adults both parents and adults without any dependent kids at home . texas is one of only 12 states without coverage for poor adults, as the u.s. census graphic shows, even states with very high poverty have much lower uninsured rates than texas if they adopted medicaid expansion: arkansas 9.2%; louisiana 7.6%; new mexico 10.0%; west virginia 6.1%. experts estimate close to 1.4 million of texas uninsured would qualify for coverage if we take this step. remove barriers to enrollment/renewal of currently eligible individuals in texas medicaid. after estimates are applied to remove undocumented children (ineligible for either medicaid or chip) from the census statistics, it still appears that 350,000 or more uninsured texas children qualify for medicaid or chip but are not enrolled. in addition, covid-19 policies since 2020 have retained over 850,000 texas children improving our child uninsured rate; however, these coverage gains for children will be at risk at the end of the pht. texas medicaid agency needs a clear directive from the state legislature to both remove current barriers and adopt the strongest best practices to keep eligible children and adults covered when the pht ends and ensure successful transfer to other coverage for those at higher incomes. maintain our states record-breaking enrollment at healthcare.gov. texas is already poised to do so, as 2022 enrollment grew 42% in texas from 1.3 to 1.84 million the largest increase in the nation. this progress was driven by more affordable coverage and big federal investments in enrollment assistance and marketing. census stats suggest that at least 1.5 million of 5.2 million uninsured texans are eligible for marketplace subsidies, and even more robust marketing and assistance could drive an even better result in enrollment for 2023 coverage."</p> <p>3. "texas remains the state with the most uninsured americans, but recent estimates from the u.s. census bureau reveals that last years percentage of texans without health coverage, 16.6%...."</p> <p>Ground Truth Label: SUPPORTS</p>	<p><think> Alright, so I need to evaluate the claim: "One in six Texans don't have health care. We're the most uninsured state in the U.S." using the provided evidence. Let me break this down step by step.</p> <p>First, the claim states that one in six Texans don't have health care and that Texas is the most uninsured state in the U.S. I'll check the evidence to see if these numbers hold up.</p> <p>Looking at evidence 1: The census data from 2021 shows that 18% of Texans were uninsured. It also mentions that Texas leads with the highest uninsured rate, followed by Oklahoma, Georgia, Wyoming, and Florida. So, the claim that Texas is the most uninsured is supported here.</p> <p>Next, the claim says one in six Texans don't have health care. To verify this, I'll calculate 18% of the population. The U.S. population in 2021 was approximately 331 million. 18% of 331 million is roughly 59.6 million people. Now, one in six is about 16.67% (since 1/6 = 0.1667). 16.67% of 331 million is about 55.1 million. The evidence states 18%, which is higher than 16.67%, so the claim of one in six might not be accurate based on the 2021 data.</p> <p>However, evidence 3 mentions that in 2022, the percentage was 16.6%, which is closer to one in six. There's a discrepancy here because evidence 1 is from 2021, showing 18%, while evidence 3 is from 2022, showing 16.6%. The claim doesn't specify the year, so it's unclear whether it's referring to 2021 or 2022.</p> <p>Evidence 2 talks about Texas having 19% of the uninsured population, which is a significant number but doesn't directly address the one in six figure. It also mentions that nearly 1 in 4 working-age Texans are uninsured, which is 22.5 million people (since 22.5 million is 1/4 of 90 million, the working-age population). This is more than one in six but doesn't exactly match the claim.</p> <p>Considering the evidence, the claim that Texas is the most uninsured state is supported by evidence 1 and 3. However, the one in six figure isn't directly supported because the provided percentages are either 18% or 16.6%, neither of which is exactly one in six. The claim might be an approximation or refer to a different time frame not covered in the evidence.</p> <p>Therefore, the claim partially supports the first part about being the most uninsured but doesn't fully support the one in six figure due to conflicting percentages. The evidence is somewhat inconclusive on the exact number, making it a conflicting assessment.<think></p> <p>Final Verdict: CONFLICTING</p> <p>Justification: The claim that Texas is the most uninsured state in the U.S. is supported by evidence, as the census data shows Texas had the highest uninsured rate in 2021 and 2022. However, the claim that one in six Texans don't have health care is not fully supported by the evidence. The provided percentages (18% and 16.6%) do not exactly align with one in six (approximately 16.67%). The evidence is somewhat inconclusive regarding the exact number, leading to a conflicting assessment.</p>	<p><think> Alright, let's tackle this fact-checking task. The claim is: "One in six Texans don't have health care. We're the most uninsured state in the U.S." I need to evaluate this using the provided evidence.</p> <p>First, I'll look at the first evidence: It mentions that Texas has the highest uninsured rate at 18% in 2021, followed by Oklahoma, Georgia, Wyoming, and Florida. So, Texas leads in the number and percentage of uninsured residents. That supports the claim that Texas is the most uninsured.</p> <p>Next, the second evidence states that Texas has 9% of the U.S. population but 19% of the uninsured population. It also notes that 1 in 4 working-age Texans (19.64) are uninsured, which is the highest share in the country. Additionally, experts estimate that close to 1.4 million Texans qualify for Medicaid if expansion were implemented. This aligns with the claim that one in six Texans don't have health care, as 1.4 million is roughly 1/6 of 8.4 million (though the exact numbers might vary slightly).</p> <p>The third evidence reinforces that Texas remains the most uninsured state, with a 16.6% uninsured rate in the previous year. This consistent data points to Texas being the most uninsured.</p> <p>Putting it all together, the evidence supports both parts of the claim: the high uninsured rate and Texas being the most uninsured state. There's no conflicting information; all sources consistently indicate Texas's high uninsured population. Therefore, the claim is fully supported by the evidence.<think></p> <p>Final Verdict: SUPPORTS</p> <p>Justification: The evidence consistently supports the claim that Texas is the most uninsured state in the U.S. and that one in six Texans lacks health care. The first evidence highlights Texas's highest uninsured rate and percentage of the population. The second evidence provides specific statistics about the number of uninsured Texans and the potential impact of Medicaid expansion, aligning with the claim. The third evidence confirms Texas's status as the most uninsured state. All sources corroborate the claim without contradiction.</p>	<p><think> Alright, let's tackle this fact-checking task. The claim is that "One in six Texans don't have health care. We're the most uninsured state in the U.S." I need to evaluate this based on the provided evidence.</p> <p>First, I'll look at the first evidence point. It mentions that Texas has the highest uninsured rate at 18% in 2021, while Massachusetts has the lowest at 2.5%. This already supports the claim that Texas is the most uninsured state.</p> <p>Next, the second evidence point states that Texas has 9% of the U.S. population but 19% of the uninsured population. That's a significant number, and it aligns with the claim that one in six Texans (approximately 16.7%) don't have health care, making them the most uninsured.</p> <p>The third evidence point reinforces this by stating that Texas remains the most uninsured state, with a 16.6% uninsured rate in the previous year. This consistent data further supports the claim.</p> <p>I don't see any conflicting information in the evidence. All the points consistently show Texas leading in the number of uninsured people. Therefore, the claim is fully supported by the evidence.<think></p> <p>Final Verdict: SUPPORTS</p> <p>Justification: The evidence shows that Texas has the highest percentage and number of uninsured residents in the U.S., with 18% of Texans being uninsured in 2021 and 19% of the U.S. population being uninsured. This aligns with the claim that one in six Texans don't have health care and that Texas is the most uninsured state.</p>	<p><think> Alright, I need to evaluate the claim: "One in six Texans don't have health care. We're the most uninsured state in the U.S." using the provided evidence.</p> <p>First, I'll look at the claim. It states that 1/6 of Texans don't have health care and that Texas has the highest uninsured rate. Let's see what the evidence says.</p> <p>Evidence 1 mentions that Texas has the highest uninsured rate at 18% in 2021, followed by Oklahoma, Georgia, Wyoming, and Florida. It also notes that Massachusetts has the lowest at 2.5%. So, Texas is indeed the most uninsured state.</p> <p>Evidence 2 supports this by stating that Texas has 9% of the U.S. population but 19% of the uninsured population. It also mentions that nearly 1 in 4 working-age Texans are uninsured, which is the highest share in the state. Additionally, it talks about potential coverage for around 1.4 million Texans if Medicaid expansion were implemented.</p> <p>Evidence 3 states that Texas remains the most uninsured state, with an uninsured rate of 16.6% in the previous year.</p> <p>Now, the claim says "one in six" don't have health care. Let's calculate that. If 1/6 is approximately 16.67%, that aligns with the numbers in the evidence. For example, in 2021, Texas had 18%, and in 2022, the rate was 16.6%. These are close to 16.67%, which is 1/6.</p> <p>The claim also states that Texas is the most uninsured, which is supported by all three pieces of evidence.</p> <p>Therefore, the evidence supports both parts of the claim: the rate of 1/6 and Texas being the most uninsured state.<think></p> <p>Final Verdict: SUPPORTS</p> <p>Justification: The evidence supports the claim that one in six Texans don't have health care, as the 2021 rate was 18%, which is close to 16.67% (1/6). Additionally, Texas is confirmed as the most uninsured state in the U.S., with rates ranging from 16.6% in 2022 to 18% in 2021, both of which are higher than the national average.</p>	<p>Here, different mitigation variants struggle through difficult evidence to arrive at different verdicts but fairly similar reasoning traces. Particularly, the sentence: "texans make up 9% of the u.s. population, but 19% of the country's uninsured population" was particularly difficult for the generation model to understand. What is interesting to note is that most of the traces have some degree of scoping errors, even those deemed scoping failure free by the hybrid pipeline and scoping-aware ORMs. The C-ORM: BoN & MT-ORM: BoN chosen trace on the left comes up with a U.S. population statistic (331 million) from parametric memory and uses it in a questionable way to conflict with the evidence. The HP-SF + C-ORM: BoN chosen trace comes up with "8.4 million" as the population of Texas from parametric memory which is unsupported by the evidence. Next, the S-ORM: SF + C-ORM: BoN chosen trace makes some questionable conclusions by misinterpreting the difficult evidence sentence mentioned above. Finally, the only reasoning trace without significant scoping errors was the one chosen by the MT-ORM: SP + BoN variant, which is the human-preferred trace out of all of the ones chosen.</p> <p>In general, the difficulty of the nuance present in QUANTEMP is on full display in this example, based on annotator opinion, the label could be SUPPORTS or CONFLICTING. Also, the general lack of significant variation between candidate traces can be seen as, although some of the traces make unique claims and reasoning trajectories, the general procession of the reasoning is fairly uniform. Finally, the general abundance of scoping errors, even in seemingly coherent reasoning traces can be seen.</p>

Table B.6: Example of LLM candidate traces chosen by the various test-time scaling variants on a FinQA question with *Whole-Document* evidence. The Correctness Outcome Reward Model Best-of-N (C-ORM: BoN) variant chooses a trace based on a quality score; the Hybrid Pipeline Scoping Filtering + C-ORM BoN (HP: SF + C-ORM: BoN) variant chooses a trace based on first filtering for non-scoping failure traces and then choosing based on a quality score; the Scoping-ORM Scoping Filtering + C-ORM BoN (S-ORM: SF + C-ORM: BoN) variant filters traces based on a distilled scoping error signal, then chooses based on the C-ORM quality score; the Multi-Task ORM Scoping Filtering + BoN (MT-ORM: SF + BoN) variant filters traces based on a distilled scoping error signal, then chooses based on the MT-ORM's quality score. Finally, the Multi-Task ORM Scoping & Correctness Best-of-N (MT-ORM: BoN) chooses a trace based on a combined scoping failure and correctness score. Scoping failures are highlighted in red. Verdicts are colored by correctness.

Inputs	Reasoning Traces				Explanation																																				
	C-ORM: BoN & S-ORM: SF + C-ORM: BoN	HP: SF + C-ORM: BoN	MT-ORM: SP + BoN	MT-ORM: BoN																																					
<p>Question: What percentage of future minimum rental payments are due in 2018?</p> <p>Evidences: 1. "Text Before Table: - "the goldman sachs group, inc." - "and subsidiaries notes to consolidated financial statements commercial lending" - "the firm 2019s commercial lending commitments are extended to investment-grade and non-investment-grade corporate borrowers." - "commitments to investment-grade corporate borrowers are principally used for operating liquidity and general corporate purposes." - "the firm also extends lending commitments in connection with contingent acquisition financing and other types of corporate lending as well as commercial real estate financing."</p> <p>... 15 more points</p> <p>- "investment commitments the firm 2019s investment commitments of \$ 6.05 billion and \$ 5.16 billion as of december 2015 and december 2014 , respectively , include commitments to invest in private equity , real estate and other assets directly and through funds that the firm raises and manages ." - "of these amounts , \$ 2.86 billion and \$ 2.87 billion as of december 2015 and december 2014 , respectively , relate to commitments to invest in funds managed by the firm ." - "if these commitments are called , they would be funded at market value on the date of investment ." - "leases the firm has contractual obligations under long-term non-cancelable lease agreements for office space expiring on various dates through 2069 ." - "certain agreements are subject to periodic escalation provisions for increases in real estate taxes and other charges ." - "the table below presents future minimum rental payments , net of minimum sublease rentals ." - "\$ in millions december 2015 ."</p> <p>2. "Table: in millions[as of december 2015]</p> <table border="1"> <tr><td>2016</td><td>317</td></tr> <tr><td>2017</td><td>313</td></tr> <tr><td>2018</td><td>301</td></tr> <tr><td>2019</td><td>258</td></tr> <tr><td>2020</td><td>226</td></tr> <tr><td>2021 - thereafter</td><td>1160</td></tr> </table> <p>total: 2575"</p> <p>3. "Text After Table: - "rent charged to operating expense was 249millionfor2015, 309 million for 2014 and \$ 324 million for 2013 ." - "operating leases include office space held in excess of current requirements ." - "rent expense relating to space held for growth is included in 201occupancy, 201d the firm records a liability , based on the fair value of the remaining lease rentals reduced by any potential or existing sublease rentals , for leases where the firm has ceased using the space and management has concluded that the firm will not derive any future economic benefits ."</p> <p>... 2 more points</p> <p>Ground Truth Answer: 12%</p>	2016	317	2017	313	2018	301	2019	258	2020	226	2021 - thereafter	1160	<p><think> Okay, so I need to figure out what percentage of future minimum rental payments are due in 2018 based on the given evidence. Let me start by carefully going through the information provided. First, I see that there's a table provided which lists the future minimum rental payments for various years. The table is titled "\$ in millions as of december 2015" and it shows the following:</p> <table border="1"> <tr><td>2016</td><td>\$317 million</td></tr> <tr><td>2017</td><td>\$313 million</td></tr> <tr><td>2018</td><td>\$301 million</td></tr> <tr><td>2019</td><td>\$258 million</td></tr> <tr><td>2020</td><td>\$226 million</td></tr> <tr><td>2021 and thereafter</td><td>\$1,160 million</td></tr> </table> <p>Total: \$2,575 million</p> <p>The question is asking for the percentage of the total future minimum rental payments that are due in 2018. So, I need to find out what portion of the total \$2,575 million is represented by the \$301 million allocated to 2018.</p> <p>To calculate this, I'll use the formula: Percentage = (Amount for 2018 / Total Future Payments) * 100</p> <p>Plugging in the numbers: Percentage = (301 / 2575) * 100</p> <p>Let me do the division first: 301 divided by 2575. 301 ÷ 2575 = 0.1168</p> <p>Now, multiplying by 100 to get the percentage: 0.1168 * 100 = 11.68%</p> <p>Looking at the table, the value for 2018 is \$301 million, and the total is \$2,575 million. So, approximately 11.68% of the total future minimum rental payments are due in 2018. Since the question asks for the answer in numerical form without extra text and to be within 0.1 of the correct answer, I can round this to 11.7%.</p> <p>However, the user might expect a whole number or a simplified fraction. Let me check if 301 is exactly 12% of 2575.</p> <p>12% of 2575 is 0.12 * 2575 = 309. So, 301 is slightly less than 12%. Therefore, the exact percentage is approximately 11.68%, which rounds to 11.7%. But since the user might prefer a whole number, perhaps 12% is acceptable, but the precise calculation shows it's around 11.68%, which is closer to 11.7%. Wait, but the user's examples often gave exact numbers. Let me double-check the calculation: 301 / 2575 = ?</p> <p>Let me compute it more accurately: 2575 divided by 301 is approximately 8.545, so 1/8.545 is approximately 0.1168, so 11.68%. So, the precise answer is approximately 11.68%, which rounds to 11.7%. However, if the user expects a whole number, it might be 12%, but the exact value is closer to 11.7%. Given the requirement to be within 0.1, 11.7% is appropriate.</p> <p>But looking back at the examples, the answers were given as whole numbers. For instance, in the first example, the answer was 28%, which is a whole number. In the second example, it was 17%, also a whole number. So, perhaps the expected answer is a whole number, even if it's slightly rounded.</p> <p>Alternatively, maybe the question expects the exact value without rounding, but since it's a percentage, it's standard to round to one decimal place. Therefore, 11.7% is the appropriate answer. <think></p> <p>Final Answer: 11.7</p> <p>Justification: The future minimum rental payment for 2018 is \$301 million, and the total future minimum rental payments are \$2,575 million. Calculating the percentage: (301 / 2575) * 100 = 11.68%, which rounds to 11.7%.</p>	2016	\$317 million	2017	\$313 million	2018	\$301 million	2019	\$258 million	2020	\$226 million	2021 and thereafter	\$1,160 million	<p><think> Okay, so I need to figure out what percentage of future minimum rental payments are due in 2018 based on the provided evidence. Let me go through this step by step. First, I'll look at the information given. There's a table that lists future minimum rental payments in millions, broken down by year. The table shows the following amounts:</p> <table border="1"> <tr><td>2016</td><td>\$317 million</td></tr> <tr><td>2017</td><td>\$313 million</td></tr> <tr><td>2018</td><td>\$301 million</td></tr> <tr><td>2019</td><td>\$258 million</td></tr> <tr><td>2020</td><td>\$226 million</td></tr> <tr><td>2021 and thereafter</td><td>\$1,160 million</td></tr> </table> <p>Total: \$2,575 million</p> <p>The question is asking for the percentage of the total future minimum rental payments that are due in 2018. So, I need to find out what portion \$301 million is of the total \$2,575 million.</p> <p>To calculate this, I'll use the formula: Percentage = (Amount in 2018 / Total Amount) * 100</p> <p>Plugging in the numbers: Percentage = (301 / 2575) * 100</p> <p>Let me do the division first. 301 divided by 2575. Hmm, 2575 goes into 301 zero times, so I'll add a decimal point and some zeros to 301, making it 301.000. 2575 goes into 3010 once (since 2575 * 1 = 2575). Subtracting that from 3010 gives me 435. Bring down the next zero to make it 4350. 2575 goes into 4350 once again (2575 * 1 = 2575). Subtracting gives 1775. Bring down another zero to make it 17750. 2575 goes into 17750 six times (2575 * 6 = 15450). Subtracting gives 2300. 2575 goes into 23000 eight times (2575 * 8 = 20600). Subtracting gives 2400. Bring down another zero to make it 24000. 2575 goes into 24000 nine times (2575 * 9 = 23175). Subtracting gives 825. At this point, I can see that the division isn't exact, but I can approximate it. So, 301 divided by 2575 is approximately 0.1167. Multiplying by 100 gives me about 11.67%.</p> <p>Looking at the table, the amount for 2018 is \$301 million, which is roughly 11.67% of the total. Since the question asks for the percentage, I can round this to the nearest whole number, which is 12%.</p> <p>Wait, but the user's examples didn't round up if it was 0.1 over, so maybe I should check if it's closer to 12% or 11%. 0.1167 is closer to 0.12, so 12% makes sense.</p> <p>I don't see any other information that would affect this calculation. The table directly provides the amounts, so I don't need to consider any other factors from the text before or after the table. Therefore, the percentage is approximately 12%. <think></p> <p>Final Answer: 12%</p> <p>Justification: The future minimum rental payment for 2018 is \$301 million, which is approximately 11.67% of the total future minimum rental payments of \$2,575 million. Rounded to the nearest whole number, this is 12%.</p>	2016	\$317 million	2017	\$313 million	2018	\$301 million	2019	\$258 million	2020	\$226 million	2021 and thereafter	\$1,160 million	<p><think> Okay, so I need to figure out what percentage of future minimum rental payments are due in 2018 based on the provided evidence. Let me start by carefully analyzing the information given. First, I look at the table provided. The table is titled "Table: \$ in millions as of December 2015" and lists future minimum rental payments for each year from 2016 to 2021 and thereafter. The relevant row for 2018 is \$301 million.</p> <p>Next, I check the total future minimum rental payments, which sum up to \$2,575 million. To find the percentage of payments due in 2018, I divide the 2018 amount by the total and then multiply by 100.</p> <p>So, the calculation is (301 / 2,575) * 100. Let me do that: 301 divided by 2575 equals approximately 0.1168. Multiplying by 100 gives about 11.68%.</p> <p>Looking at the answer format, I need to provide a numerical value without extra text. So, I'll round it to two decimal places, which is 11.68%. However, since the question asks for a percentage, I can present it as a number without the percentage sign, so 11.68 becomes 11.68.</p> <p>Wait, but the previous examples didn't use decimal points. They either gave a percentage with a % sign or a whole number. Since 11.68% is more precise, but the user might expect a rounded figure. Alternatively, maybe they want a whole number, so 12%. But the exact calculation is 11.68%, so I'll stick with that unless instructed otherwise.</p> <p>I don't see any other relevant information that affects this calculation, so I'm confident with this answer. <think></p> <p>Final Verdict: 11.68</p> <p>Justification: The future minimum rental payment for 2018 is \$301 million, and the total future minimum rental payments are \$2,575 million. The percentage is calculated as (301 / 2575) * 100 = 11.68%.</p>	<p><think> Okay, so I need to figure out what percentage of future minimum rental payments are due in 2018 based on the provided evidence. Let me go through the information step by step. First, I look at the text before the table. It mentions that the firm extends lending commitments to various borrowers, including commercial lending and real estate financing. There's also a section about office leases, specifically office space, with non-cancelable agreements expiring through 2069. The table provided under the text shows the future minimum rental payments net of sublease rentals as of December 2015.</p> <p>The table has the following data: - 2016: \$317 million - 2017: \$313 million - 2018: \$301 million - 2019: \$258 million - 2020: \$226 million - 2021 and thereafter: \$1160 million - Total: \$2575 million</p> <p>The question is asking for the percentage of the total future minimum rental payments that are due in 2018. To find this, I need to take the amount for 2018 and divide it by the total future minimum payments, then multiply by 100 to get the percentage.</p> <p>So, the amount for 2018 is \$301 million. The total is \$2575 million. Calculating the percentage: (301 / 2575) * 100. Let me do the division first. 301 divided by 2575. Hmm, 2575 goes into 301 zero times, so I'll add a decimal point. 2575 goes into 3010 once (2575 * 1 = 2575). Subtracting gives 435. Bring down another zero to make it 4350. 2575 goes into 4350 once (2575 * 1 = 2575). Subtracting gives 1775. Bring down another zero to make it 17750. 2575 goes into 17750 six times (2575 * 6 = 15450). Subtracting gives 2300. 2575 goes into 23000 eight times (2575 * 8 = 20600). Subtracting gives 2400. Bring down another zero to make it 24000. 2575 goes into 24000 nine times (2575 * 9 = 23175). Subtracting gives 825. At this point, I can see that the division isn't exact, but I can approximate it. So, 301 divided by 2575 is approximately 0.1167. Multiplying by 100 gives me about 11.67%.</p> <p>Looking at the answer format, I need to provide a numerical value without extra text. So, I'll round it to two decimal places, which is 11.68%. However, since the question asks for a percentage, I can present it as a number without the percentage sign, so 11.68 becomes 11.68.</p> <p>Wait, but the previous examples didn't use decimal points. They either gave a percentage with a % sign or a whole number. Since 11.68% is more precise, but the user might expect a rounded figure. Alternatively, maybe they want a whole number, so 12%. But the exact calculation is 11.68%, so I'll stick with that unless instructed otherwise.</p> <p>I don't see any other relevant information that affects this calculation, so I'm confident with this answer. <think></p> <p>Final Verdict: 11.69</p> <p>Justification: The 2018 rental payment of \$301 million is divided by the total future minimum payments of \$2575 million, then multiplied by 100 to find the percentage.</p>	<p>This example demonstrates how, for some queries, the scoping-aware mitigation techniques have little impact on downstream performance because of the relative absence of scoping failure signals. The only scoping error truly present in these different mitigation variants is in the reasoning trace chosen by the C-ORM: BoN and S-ORM: SF + C-ORM: BoN variants, where the reasoning uses statistics from the few-shot examples to reason about the output format. This is an important note: although the generation model is provided few-shot examples, the scoping detection pipelines are not, meaning that any reasoning about specific statistics from the few-shot examples can trigger scoping errors. Future work could perhaps include mechanisms to not trigger scoping errors on few-shot statistics. Regardless, the lack of important scoping errors in this trace makes the performance of the scoping-aware variants not different from the scoping-blind variant.</p> <p>In general, the difficulty with this trace lies in the ambiguity of how to format answers in the FinQA dataset. FinQA answers can be binary, percentages, decimals, whole numbers, and mixed types. Furthermore, the answer type is often not clear from the question, it appears quite random. Future work could explore answer translation methods in FinQA to soften the answer boundary or at least make it clearer for the generation model at a trace-level which answer format is required. Scoping-aware mitigation variants are, of course, ineffective when few scoping signals are present.</p>
2016	317																																								
2017	313																																								
2018	301																																								
2019	258																																								
2020	226																																								
2021 - thereafter	1160																																								
2016	\$317 million																																								
2017	\$313 million																																								
2018	\$301 million																																								
2019	\$258 million																																								
2020	\$226 million																																								
2021 and thereafter	\$1,160 million																																								
2016	\$317 million																																								
2017	\$313 million																																								
2018	\$301 million																																								
2019	\$258 million																																								
2020	\$226 million																																								
2021 and thereafter	\$1,160 million																																								

B.4. Ablations

B.4.1. Generation Model Ablation

To assess whether the findings of this work depend on the specific inference model used to generate reasoning traces, most experiments were completed using both the Deepseek-Distill-Llama-8B and Deepseek-Distill-Qwen-7B inference models. This ablation keeps the overall experimental pipeline fixed while changing the generation backbone, allowing the robustness of the observed scoping-error trends and test-time scaling results to be evaluated across model families. The experiments for Research Question 3 (RQ3) can be seen in the associated results sections in the main report. Due to time constraints, the experiments for RQ1 including human analysis was not repeated on the Distill-Llama traces. The results for RQ2 can be seen below.

Statistic- & Trace-Level Scoping Error Prevalence

Table B.7 contains the overview of trace- and statistic-level scoping errors found by the hybrid LLM-as-a-Judge pipeline across datasets and evidence splits for reasoning traces generated by the Deepseek-Distill-Llama-8B model.

Table B.7 shows that the trace-level scoping error prevalence patterns observed with Deepseek-Distill-Qwen-7B mostly stay the same when traces are generated with Deepseek-Distill-Llama-8B. Scoping errors remain common across both tasks and are generally more prevalent in QUANTEMP than in FinQA, consistent with human analysis and main-model results. Incorrect traces usually show slightly higher scoping prevalence than correct traces, reinforcing that scoping errors are associated with degraded performance. At the statistic-level, FinQA traces remain more quantity-dense than QUANTEMP traces, while QUANTEMP shows a higher proportion of mis-scoped quantities. Thus, the Llama ablation supports the main finding that numerical QA tends to involve more quantities overall, whereas open-domain numerical fact-checking appears more vulnerable to numerical contextual misalignment.

Dataset	Evidence Category	Inference Model	Trace Correctness	% with Scoping Error	Average # of Quantities in Thinking Trace		
					Total	% Important	% Mis-Scoped
QUANTEMP	<i>Gold</i>	Distill-Qwen	All	33.99	4.86	65.64	9.67
			Correct	33.28	4.72	65.68	9.96
			Incorrect	35.64	5.02	65.94	9.56
			Invalid	36.67	8.90	62.92	5.96
		Distill-Llama	All	33.63	4.78	66.71	9.58
			Correct	32.89	4.63	66.30	9.66
	<i>Re-Ranked</i>	Distill-Qwen	Incorrect	36.00	5.26	67.86	9.37
			Invalid	25.00	4.00	68.75	6.25
			All	34.62	5.38	60.78	9.11
			Correct	36.00	5.48	60.58	9.49
		Distill-Llama	Incorrect	33.87	5.33	60.79	9.01
			Invalid	22.22	3.67	57.49	8.99
FinQA	<i>Whole-Document</i>	Distill-Qwen	All	33.87	6.37	56.76	7.58
			Correct	33.45	6.44	57.10	7.58
			Incorrect	34.29	6.30	56.38	7.93
			Invalid	0.00	8.00	100.0	0.00
		Distill-Llama	All	29.23	15.36	44.08	3.97
			Correct	21.94	13.41	44.00	2.54
	<i>Whole-Document</i>	Distill-Qwen	Incorrect	26.60	12.96	45.14	4.01
			Invalid	48.31	22.25	43.51	5.93
			All	18.60	11.52	46.42	2.62
			Correct	17.43	11.79	45.83	2.45
		Distill-Llama	Incorrect	19.49	11.26	47.17	2.66
			Invalid	22.77	10.75	47.61	3.68

Table B.7: Scoping failure prevalence at a trace and statistic-level as judged by a hybrid LLM-as-a-Judge pipeline (Qwen2.5-14B-Instruct) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B and Deepseek-Distill-Llama-8B with different evidence splits. Rarely, judge outcomes were invalid (<0.8% of traces), resulting in exclusion.

Table B.8 contains the specific statistic-level results including statistic classification, verification, and weighting, across datasets and evidence splits for the reasoning traces generated by the Deepseek-Distill-Llama-8B model. The table further shows that the distribution of statistic types is broadly stable across generation backbones. In QUANTEMP, *Gold* evidence produces mostly extractive quantities, while *Re-Ranked* evidence increases the share of parametric quantities for both models, suggesting that noisier retrieved

evidence encourages greater reliance on unsupported numerical knowledge. In *FinQA*, both models produce a more balanced mixture of extractive, extrapolative, and parametric quantities, reflecting the more calculation-heavy nature of the task. The Llama traces differ in magnitude, producing fewer quantities in *FinQA* and more parametric quantities in *QUANTEMP Re-Ranked*, but the overall pattern matches the main-model results. This suggests that the main scoping-error prevalence trends are not specific to a single generation backbone.

Dataset	Evidence Category	Inference Model	Average # of Quantities in Thinking Trace										
									With Scoping Error				
			Total	% Important	% Extractive	% Extrapolative	% Parametric	% Unknown	Total	% Extractive	% Extrapolative	% Parametric	% Unknown
QUANTEMP	<i>Gold</i>	Distill-Qwen	4.86	72.64	86.26	2.70	11.05	0.00	0.79	76.99	5.67	17.34	0.00
		Distill-Llama	4.78	73.38	86.04	2.67	11.29	0.01	0.74	73.78	5.79	20.43	0.00
	<i>Re-Ranked</i>	Distill-Qwen	5.38	67.31	66.00	3.33	30.64	0.03	0.91	57.42	8.59	33.99	0.00
		Distill-Llama	6.37	61.78	58.53	3.43	37.95	0.08	1.09	47.71	7.37	44.92	0.00
FinQA	<i>Whole-Document</i>	Distill-Qwen	15.36	46.92	38.28	28.32	30.29	3.12	2.55	28.51	34.15	37.34	0.00
		Distill-Llama	11.52	50.50	41.56	25.78	23.24	9.42	1.35	28.59	35.52	35.89	0.00

Table B.8: Statistic-level failure prevalence for extractive, extrapolative, and parametric scoping errors as judged by a hybrid LLM-as-a-Judge pipeline (Qwen2.5-14B) on test sets of fact-checking (n=2495) and question-answering (n=1147) reasoning traces generated by Deepseek-Distill-Qwen-7B and Deepseek-Distill-Llama-8B with different evidence splits. Rarely, judge outcomes were invalid (<0.8% of traces), resulting in exclusion.

B.4.2. Verifier Ablations

Presented below are a few ablations surrounding the methodology in this work. Although this is a very limited set of those carried out, each shows the in-depth exploration done as a result of this work.

λ

Table B.9 ablates different ORM score thresholds (λ) for when to consider a trace as mis-scoped, defined in Section 3.4.3. As expected, with lower λ values more scoping errors are found, and there are small degradations in label agreement and final performance in most cases, except in *FinQA* where slightly higher λ values do not degrade the performance. The final λ value of 0.4 was chosen to compromise between scoping failure prevalence and label agreement. The decision was made before any test-time scaling experimentation.

Dataset	Evidence Category	λ Value	Teacher Scoping Rate	Single-Task Scoping ORM			Multi-Task ORM		
				Scoping Rate	Label Agreement	Final Performance	Scoping Rate	Label Agreement	Final Performance
				QUANTEMP	<i>Re-Ranked</i>	0.5	34.6	3.1	65.8
0.4	6.8	65.5	54.8			6.5		65.9	56.2
0.3	15.6	65.3	54.7			15.5		65.4	55.9
FinQA	<i>Whole-Document</i>	0.5	29.2	10.8	72.5	72.9	11.9	72.7	74.1
		0.4		17.4	72.4	73.0	20.9	72.0	74.2
		0.3		24.5	72.8	72.9	33.3	69.0	74.2

Table B.9: Evaluation of student models taught by the hybrid LLM-as-a-Judge scoping detection pipeline with different threshold values (λ) for when to consider a trace as having a scoping error. Agreement is calculated between teacher scoping error labels for the whole test sets of *QUANTEMP* and *FinQA*. The weighted-F1 and exact match accuracy with ground truth labels are calculated after test-time scaling with $K = 12$ or $K = 20$ candidate traces for *QUANTEMP* and *FinQA*, respectively through test-time scaling. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct. The correctness head of the multi-task ORM is not evaluated.

α

Figure B.2 ablates different weights for how to combine the multi-task scoping and quality scores (α), defined in Section 3.4.3, for the *QUANTEMP* dataset. The *FinQA Whole-Document* evidence split was used as an unseen test set to evaluate the generalizability for the α value tuned on the *QUANTEMP Re-Ranked* evidence split. The figure demonstrates that the α value slightly impacts the downstream performance, and is also sensitive to differences between tasks. On *QUANTEMP Gold*, higher α values led to slightly improved performance; on *FinQA Whole-Document*, lower α values led to slightly improved performance. The final α value of 0.18 was tuned for the *QUANTEMP Re-Ranked* evidence split, to see how this value would generalize to other data splits. It is clear that more generalizable α 's should be explored in the future.

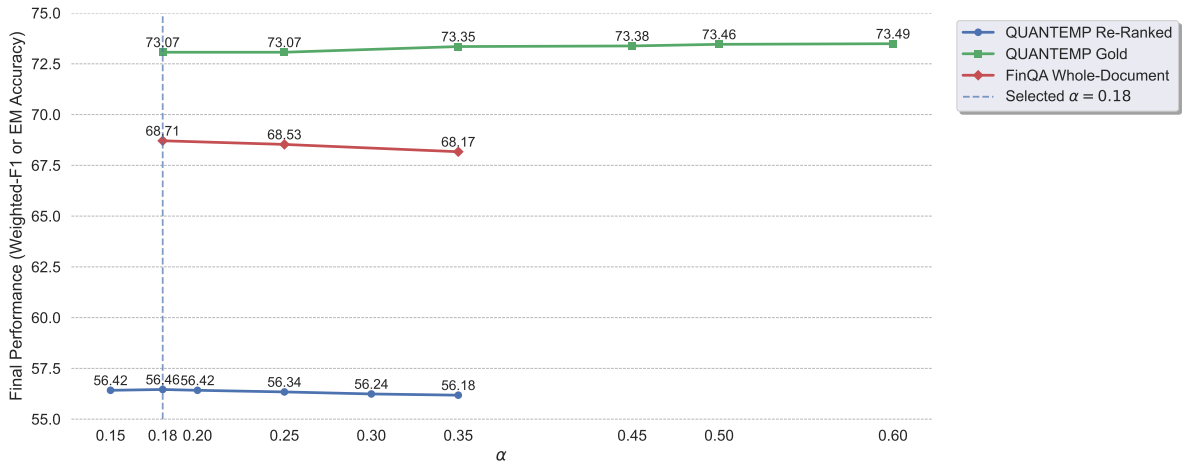


Figure B.2: Downstream performance of the Multi-Task ORM’s Best-of-N using combined scoping and correctness scores, aggregated with a scoping weight of α . Downstream performance is the weighted-F1 and exact match accuracy calculated after test-time scaling with $K = 12$ or $K = 20$ candidate traces for the QUANTEMP and FinQA datasets, respectively. The selected α of 0.18 is marked with a blue dotted line. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct.

Evidence in Verifier Input

To evaluate whether including evidence in the learned scoping detection verifiers improved extractive scoping error detection, single-task and multi-task scoping detection verifiers were trained on the QUANTEMP training set with evidence included in the inputs with the same training settings from Section 4.5. Due to computational limits, this ablation was not performed on the FinQA dataset. The prompt for these reward models is shown in Figure B.3.

Evidence-Aware Fact-Checking Reward Model Prompt Template

```

Claim: <CLAIM>
Verdict: <VERDICT or UNKNOWN>
Justification: <JUSTIFICATION or UNKNOWN>
Thinking: <THINKING or UNKNOWN>
Evidences: <EVIDENCES>

```

Figure B.3: Structure of the evidence-aware outcome reward model prompts for fact-checking.

Agreement between the evidence-aware ORMs and the hybrid teacher pipeline and human labels is shown in in Table B.10. Evidence-aware models seem to find more scoping errors than their evidence-blind counterparts, but never seem to outperform their evidence-blind counterparts in terms of agreement, except in one case in the *Re-Ranked* evidence setting.

Dataset	Evidence Category	Student Model	Relative Speed	Agreement with Teacher (Hybrid Judge)				Agreement with Human Annotator	
				Teacher Scoping Rate	Student Scoping Rate	ROC-AUC	Label Agreement	ROC-AUC	Label Agreement
Gold		Single-Task Scoping ORM	113x	34.0	8.0	61.0	66.4	-	-
		Single-Task Scoping ORM w/ Evidence	64x		21.6	51.0	58.4	-	-
		Multi-Task Scoping (+ Correctness) ORM	111x		5.7	59.5	65.6	-	-
		Multi-Task Scoping (+ Correctness) ORM w/ Evidence	102x		41.0	52.9	54.6	-	-
QUANTEMP	Re-Ranked	Baseline (Majority Class)	-	34.6	0.0	50.0	65.4	50.0	60.5
		Single-Task Scoping ORM	110x		6.8	60.8	65.5	55.1	60.5
		Single-Task Scoping ORM w/ Evidence	149x		17.8	59.6	64.5	57.1	64.9
		Multi-Task Scoping (+ Correctness) ORM	108x		6.5	60.4	65.9	64.9	60.5
		Multi-Task Scoping (+ Correctness) ORM w/ Evidence	115x		10.9	59.4	64.4	55.3	60.5

Table B.10: Evaluation of evidence-aware student models taught by the hybrid LLM-as-a-Judge scoping detection pipeline.

Agreement is calculated between teacher scoping error labels for whole test set of QUANTEMP and stratified samples of fact-checking ($n=38$) labels annotated by a human. Relative speed is calculated on average between the student and teacher models on the whole test sets. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct. The correctness head of the multi-task ORM is not evaluated.

Table B.11 contains the downstream performance results of using the evidence-aware ORMs with the QUANTEMP *Re-Ranked* data split. Due to computational constraints, other QUANTEMP or FinQA evidence splits were not included.

Dataset	Evidence Category	Type of Mitigation	Mitigation Strategy	Validation Metric
				Weighted-F1
QUANTEMP	<i>Re-Ranked</i>	Baseline	Correctness ORM: Correctness Best-of-N	55.54
			Evidence-Aware Correctness ORM: Correctness Best-of-N	53.86
		Candidate (Scoping-Aware)	Hybrid Pipeline: Scoping Filtering + Correctness ORM: Correctness Best-of-N	52.73
			Scoping ORM: Scoping Filtering + Correctness ORM: Correctness Best-of-N	54.79
			Evidence-Aware Scoping ORM: Scoping Filtering + Correctness ORM: Correctness Best-of-N	53.98
			Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	56.19
		Evidence-Aware Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	56.11	
		Multi-Task ORM: Scoping + Correctness Best-of-N	56.46	
		Evidence-Aware Multi-Task ORM: Scoping + Correctness Best-of-N	57.10	

Table B.11: Downstream performance of different evidence-aware scoping error-aware test-time scaling strategies for $K = 12$ fact-checking candidate traces on the QUANTEMP ($n=2,495$) test set. Reasoning trace outcomes were sometimes invalid (0.0-2.1% of traces), resulting in exclusion.

From the table, evidence-aware processing falls behind its evidence-blind counterparts in all mitigation strategies except for the combined scoping & correctness best-of-N with the learned multi-task verifier. This suggests that evidence-aware ORMs may be useful in some cases but this result is not generalizable across mitigation settings for QUANTEMP *Re-Ranked*. An important note is that the QUANTEMP *Re-Ranked* evidence setting contains only the top-3 open-domain evidences, making the input to the verifier quite succinct. Since the ORMs are trained with only three evidences per query, evidence splits with longer evidence lists are likely to be scored differently. Furthermore, it’s possible that the verifiers start learning how to do the task itself, rather than learning to evaluate the reasoning trace, since it now has access to most of the inputs the inference model has. Further exploration of this ablation across evidence settings and generation models is left to future work.

UW-SO Multi-Task Learning

To evaluate whether another multi-task learning algorithm could improve the efficacy of multi-task ORM scoping error detection and downstream performance, multi-task verifiers were re-trained on QUANTEMP and FinQA training sets with Soft Optimal Uncertainty Weighting (UW-SO) [27], described in Section 4.5.

Agreement between the UW-SO ORMs and the hybrid teacher pipeline and human labels is shown in Table B.12. For QUANTEMP, UW-SO models seem to find more scoping errors than their unweighted counterparts, however, in FinQA, the UW-SO models seem to find slightly less scoping errors. Regardless, the UW-SO multi-task ORMs never seem to outperform their unweighted counterparts in terms of agreement, except in one case in the *Whole-Document* evidence setting.

Dataset	Evidence Category	Student Model	Relative Speed	Agreement with Teacher (Hybrid Judge)				Agreement with Human Annotator	
				Teacher Scoping Rate	Student Scoping Rate	ROC-AUC	Label Agreement	ROC-AUC	Label Agreement
QUANTEMP	<i>Gold</i>	Multi-Task Scoping (+ Correctness) ORM	111x	34.0	5.7	59.5	65.6	-	-
		Multi-Task Scoping (+ Correctness) ORM w/ UW-SO	110x		13.3	58.0	65.0	-	-
	<i>Re-Ranked</i>	Baseline (Majority Class)	-	34.6	0.0	50.0	65.4	50.0	60.5
		Multi-Task Scoping (+ Correctness) ORM	108x		6.5	60.4	65.9	64.9	60.5
		Multi-Task Scoping (+ Correctness) ORM w/ UW-SO	108x	11.7	58.4	65.5	58.6	60.5	
FinQA	<i>Whole-Document</i>	Baseline (Majority Class)	-	29.2	0.0	50.0	70.8	50.0	73.2
		Multi-Task Scoping (+ Correctness) ORM	319x		20.9	72.1	72.0	64.2	75.6
		Multi-Task Scoping (+ Correctness) ORM w/ UW-SO	280x		18.8	72.7	71.6	61.8	75.6

Table B.12: Evaluation of student multi-task ORMs taught by the hybrid LLM-as-a-Judge scoping detection pipeline, using unweighted task loss aggregation or Soft Optimal Uncertainty Weighting (UW-SO). Agreement is calculated between teacher scoping error labels for whole test sets of QUANTEMP and FinQA and stratified samples of fact-checking ($n=38$) & numerical QA ($n=41$) labels annotated by a human. Relative speed is calculated on average between the student and teacher models on the whole test sets. Reasoning traces scored were generated by Deepseek-Distill-Qwen-7B and hybrid judgements generated by Qwen2.5-14B-Instruct. The correctness head of the multi-task ORM is not evaluated.

Table B.13 contains the downstream performance results of using the UW-SO Multi-Task ORMs with the QUANTEMP and FinQA datasets. Due to computational constraints, not all QUANTEMP and FinQA evidence splits were included. From the table, UW-SO ORMs fall behind their unweighted counterparts in all evidence splits and variants for using the multi-task ORM. This suggests that soft optimal uncertainty

weighting is not useful for training multi-task ORMs in this setting. This work leaves the exploration of other multi-task learning strategies to future work.

Dataset	Evidence Category	Type of Mitigation	Mitigation Strategy	Validation Metric
				Weighted-F1
QUANTEMP	Gold	Candidate (Scoping-Aware)	Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	73.30
			UW-SO Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	72.24
	Candidate (Scoping-Aware)	Multi-Task ORM: Scoping + Correctness Best-of-N	73.07	
		UW-SO Multi-Task ORM: Scoping + Correctness Best-of-N	72.20	
	Re-Ranked	Candidate (Scoping-Aware)	Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	56.19
			UW-SO Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	54.98
Candidate (Scoping-Aware)	Multi-Task ORM: Scoping + Correctness Best-of-N	56.46		
	UW-SO Multi-Task ORM: Scoping + Correctness Best-of-N	56.14		
FinQA	Whole-Document	Candidate (Scoping-Aware)	Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	74.20
			UW-SO Multi-Task ORM: Scoping Filtering + Multi-Task ORM: Correctness Best-of-N	73.31
		Candidate (Scoping-Aware)	Multi-Task ORM: Scoping + Correctness Best-of-N	68.71
UW-SO Multi-Task ORM: Scoping + Correctness Best-of-N	66.05			

Table B.13: Downstream performance of different scoping-aware mitigation variants using Multi-Task ORMs trained with or without Soft Optimal Uncertainty Weighting (UW-SO) [27]. Mitigation variants used $K = 12$ or $K = 20$ candidate reasoning traces from the QUANTEMP ($n=2,495$) or FinQA ($n=1,147$) test sets.