UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

Trustworthy Language from Structured Evidence: Explanations, Taxonomies, and Diagnostics

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

BY

Lena Tannaz Trigg
Norman, Oklahoma
2026

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/342685

TRUSTWORTHY LANGUAGE FROM STRUCTURED EVIDENCE:
EXPLANATIONS, TAXONOMIES, AND DIAGNOSTICS

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY THE COMMITTEE CONSISTING OF

Dr. Dean Frederick Hougen, Chair

Dr. Dimitrios Diochnos

Dr. Chao Lan

Dr. Talayeh Razzaghi

© Copyright by Lena Tannaz Trigg 2026
All Rights Reserved.

Acknowledgements

I would like to express my deepest gratitude to the many individuals who have supported me throughout my doctoral journey.

First and foremost, I extend my sincere thanks to my research advisor, Dr. Dean F. Hougen, for his exceptional supervision, insightful guidance, encouragement, and patience. His mentorship not only shaped the direction of my research but also made my doctoral experience truly rewarding. I am especially grateful for his kindness and unwavering support during challenging times. I could not have asked for a better advisor.

I am also deeply thankful to my doctoral committee members, Dr. Dimitrios Diochnos, Dr. Chao Lan, and Dr. Talayeh Razzaghi, for their time and support of my work. My gratitude extends to the University of Oklahoma (OU), Dr. Sridhar Radhakrishnan, and Dr. Dean Hougen for providing continuous graduate assistantship positions that made my doctoral studies possible.

I would also like to thank the Graduate College and Dr. Sherri Irvin for their guidance, support, and understanding, particularly during the most difficult transitions of my doctoral journey.

My heartfelt appreciation goes to my family, my father Mehdi Sadri and my mother Firoozeh Barkhordar, for their unconditional love, sacrifices, and unwavering belief in me. I owe them everything I have achieved. I am also deeply

thankful to my beloved sister Sadaf Sadri for her endless encouragement throughout my education. This dissertation fulfills not only my own aspiration but also the cherished dream of my late grandmother, Badr Mohammadzadeh, and my parents.

To my husband and best friend, Scott Trigg, I extend my deepest love and gratitude for his constant support, patience, and understanding. His companionship, encouragement, and thoughtful insights have been invaluable throughout my research and writing.

Finally, I would like to express a special thank you to my dear friend Delaram Nemattollahi for her kindness, positivity, and support at all times.

To all of you, thank you for being part of this journey.

Abstract

Emerging AI systems must not only act correctly but also explain their behavior in verifiable natural language. This dissertation traces a research arc that begins with NavChat, a post-hoc framework for explaining Uncrewed Aerial Vehicle (UAV) navigation decisions made by deep reinforcement learning (DRL) policies. While NavChat's large language model (LLM) rationales are often fluent and intuitive, they reveal systematic reasoning failures, such as mis-handling negative deltas, demonstrating that fluency alone is not evidence of faithfulness and raising a broader question: *Do current models reliably perform operator-level reasoning over structured evidence, and how can we measure it?*

Motivated by the structured, episodic nature of UAV navigation traces, which can be represented as tabular sequences, we pivot to the Logical Table-to-Text (LT2T) paradigm, where tables provide a controllable sandbox for probing logical fidelity. First, we contribute a comprehensive LT2T survey that codifies core challenges, method taxonomies, and existing gaps, including weak evaluation metrics. **We find that prior evidence for *strong* LLM reasoning is often based on aggregate automatic scores that obscure which logical operations models actually execute correctly and where verifiers disagree.** Building on this finding, we design an operation-aware diagnostic framework for LLM-based LT2T that decomposes performance into four competencies: (1) logical form ex-

ecution, (2) logical form conditioned generation, (3) logic type prediction, and (4) logical form free generation. This framework reveals a persistent meta-logical gap: models can produce plausible or even logically entailed statements while failing to identify the intended underlying operation, such as aggregation versus ordinal reasoning.

This dissertation makes three contributions: (i) NavChat, an LLM-assisted method for rationalizing DRL navigation decisions and empirically characterizing explanation failure modes; (ii) a Logical Table-to-Text survey that systematizes methods and limitations for reasoning-grounded generation; and (iii) an operation-aware diagnostic framework that localizes model and evaluator failures by logical competency. Together, these contributions advance trustworthy language interfaces for structured AI by grounding explanations in model-relevant evidence and providing transparent diagnostics that clarify not only whether they fail, but also why they fail.

Contents

Acknowledgements **iv**

Abstract **vi**

List of Figures **ix**

List of Tables **x**

1 Introduction **1**

2 Natural Language Explanation for Deep Reinforcement Learning (DRL)-based Navigation **6**
2.1 Introduction 7
2.2 Background 9
2.2.1 Reinforcement Learning 9
2.2.2 Large Language Models as Reasoners 11
2.3 Related Work 12
2.3.1 Explainable Deep Reinforcement Learning 12
2.3.2 Explainable DRL-Based Aerial Navigation 13
2.4 Methodology 13
2.5 Experiments and Results 17
2.5.1 DRL Navigation system and environment 17
2.5.2 Text Generation Results 18
2.6 Conclusions and Future Work 29

3 Logical Table-to-Text Generation: Challenges, Methods, and Reasoning **31**
3.1 Introduction 32
3.2 Background 33
3.2.1 Logical Table-to-Text 33
3.2.2 Reasoning 34
3.3 Datasets 35
3.4 Models 37

3.5 Challenges in LT2T . . . 39
3.6 Methods for Logical Table-to-Text . . . 41
3.6.1 Logical Fidelity and Controllability . . . 41
3.6.2 Numerical Reasoning . . . 46
3.6.3 Data Scarcity . . . 49
3.6.4 Diversity . . . 50
3.6.5 User Preference . . . 50
3.6.6 Evaluation Methodology . . . 50
3.7 Comparing Methods . . . 51
3.7.1 Other evaluation metrics . . . 53
3.8 Datasets and Evaluation Metrics . . . 57
3.9 Discussion . . . 57
3.10 Future Directions . . . 58
3.11 Conclusions . . . 60

4 Operation-Level Logical Fidelity in Table-to-Text Generation 61
4.1 Introduction . . . 62
4.2 Related Work . . . 64
4.3 Task Setup and Problem Formulation . . . 65
4.3.1 Logical Table-to-Text and Logical Forms . . . 65
4.3.2 Logic Type Taxonomy . . . 65
4.3.3 Task Definitions . . . 66
4.3.4 What These Tasks Enable . . . 67
4.4 Experiments . . . 68
4.4.1 Test Set . . . 68
4.4.2 Automated Logical Fidelity Metrics . . . 68
4.4.3 Evaluation Results . . . 68
4.5 Discussion . . . 86
4.5.1 Metric Disagreement Analysis. . . . 86
4.5.2 Qualitative Error Analysis . . . 89
4.6 Conclusions . . . 90

5 Conclusions 91

6 Future Work 94

Bibliography 97

A Appendix 106
A.1 Logic Type . . . 106
A.2 Taxonomy of error type . . . 108

List of Figures

1.1 Overall structure of the dissertation (AI-generated). 3

2.1 NavChat Framework 14
2.2 Common part of all prompts (Role). 15
2.3 Zero-shot prompt. 15
2.4 One-shot prompt. 15
2.5 Sample of Chain of Thought prompt 16
2.6 State information of two consecutive time steps. 21
2.7 Evaluation and justification generated by GPT-3.5. 22
2.8 Evaluation and justification generated by GPT-4o. 22
2.9 The trajectory of the scenario in the user study 26
2.10 Plots demonstrating the overall agreements in two tasks. 28
2.11 Plots demonstrating the extent of the agreements with NavChat's explanation for each question in two tasks. 29
2.12 Ranking of various methods of explanations. 29

3.1 Comparing surface and logical-level outputs. 32
3.2 Taxonomy of logical table-to-text methods 41

4.1 Operation-aware diagnostic framework 66
4.2 Confusion matrix heatmap of all models. 82

List of Tables

2.1 The accuracy of different models in evaluating the actions using different prompting methods. 20
2.2 Rules used to determine effectiveness, alignment, and safety before assigning the final action label. 24
2.3 NavChat action-category mapping. 24
2.4 Questions used in the user study 26

3.1 Logical Table-to-Text Datasets. 35
3.2 Traditional encoder-decoders trained from scratch for LT2T. MLE = Maximum Likelihood Estimation, RL = Reinforcement Learning, CI = Causal Intervention, Trans = Transformer 37
3.3 Pre-trained language models fine-tuned for LT2T. MLE = Maximum Likelihood Estimation, Adv-Reg = Adversarial Regularization, RL = Reinforcement Learning, PT = Pre-training, FT = Fine-tuning. 38
3.4 LLMs used via prompting for LT2T. 38
3.5 Comparison of LT2T Models on Logical Fidelity Metrics. Shows the best results reported in the original papers. 52
3.6 Comparison of LT2T Models on other logical fidelity Metrics. . . . 54
3.7 Comparing methods based on various criteria. 55
3.8 Datasets and evaluation metrics used in surveyed papers. 57

4.1 Task 1 results: model accuracy on logical-form execution. Columns are ordered by ascending overall average across logic types. 69
4.2 Most common LF execution failure patterns aggregated across frontier models, grouped by logic type. 71
4.3 Automated logical fidelity evaluation of generated sentences on Logic2Text with given LFs. 73
4.4 Automated logical fidelity evaluation of generated sentences on Logic2Text with given LFs. 74
4.5 Family comparison using macro-average scores across evaluator metrics. 77
4.6 LF-adherence investigation by a human study. 79

4.7 Accuracy of models in logic type prediction 81
4.8 Task 4: automated logical-fidelity evaluation on Logic2Text *without* provided LFs (higher is better). 84
4.9 Task 4: SP-Acc on Logic2Text *without* LFs 85
4.10 Average pairwise disagreement rates across logic types (over models). 86
4.11 Pairwise disagreement across metric pairs 87
4.12 Directional analysis on disagreement-sampled claims. 89

A.1 LT2T Logic Types 107
A.2 Failure patterns for Aggregation and Comparative logic types. . . 108
A.3 Failure patterns for Count and Majority logic types. 109
A.4 Failure patterns for Ordinal, Superlative, and Only logic types. . . 110

Chapter 1

Introduction

Emerging AI systems must not only act correctly but also *explain* their behavior in natural language that is faithful to the underlying evidence and computation. This need is especially acute in safety-critical settings such as autonomous aerial navigation, scientific reporting, and decision-support dashboards, where fluent text can mask subtle but consequential reasoning errors. However, a significant gap has emerged: the very models used to generate such explanations are often more fluent than faithful, creating a risk that human operators will be misled by articulate but unsupported justifications. Recent progress in deep learning and large language models (LLMs) has made it feasible to generate persuasive explanations and summaries. However, ensuring that such explanations are *verifiable*—that is, grounded in the underlying data and computation—remains an open challenge. We argue that fluent language over structured evidence is not sufficient; trustworthy systems require operation-level diagnostics that reveal whether reasoning is faithful and where it fails. Building on this idea, this dissertation contributes to the research agenda of trustworthy language interfaces for structured AI by developing and applying an operation-aware diagnostic lens.

This lens is used to evaluate when LLM-produced text is faithful to structured evidence and to localize why it fails.

This dissertation traces a research arc that begins in embodied autonomy and leads to operation-level diagnostics for structured data generation. In our navigation setting, a deep reinforcement learning (DRL)-based Uncrewed Aerial Vehicle (UAV) selects actions from numerical state features and reward signals, forming a highly structured decision trace. We introduced NavChat, a post-hoc explanation framework that utilizes an LLM to generate natural-language rationales for the agent's actions. While NavChat improves the interpretability of a complex system for non-expert users, our evaluation revealed systematic reasoning failures in LLM rationales, such as mis-handling negative deltas, even when the text appeared coherent. These observations motivate a broader question: ***Do current models possess reliable operation-level reasoning over structured evidence, and how can we measure and diagnose that reliability?*** If language is to serve as a trustworthy interface between structured AI systems and human users, explanations must be grounded in evidence, and reasoning failures must be diagnosable at the level of individual operations. Such a diagnosis creates a useful feedback loop: better evaluations reveal concrete failure modes, which in turn guide the design of more faithful and reliable systems.

Toward this goal, this dissertation develops three complementary contributions, depicted in Figure 1.1, that bridge explanation in navigation with reasoning evaluation in structured generation:

1. **Rationalizing the decisions of opaque navigation policies (NavChat).** We propose NavChat, a post-hoc natural-language explanation framework for DRL-based UAV navigation. NavChat uses model-relevant signals,

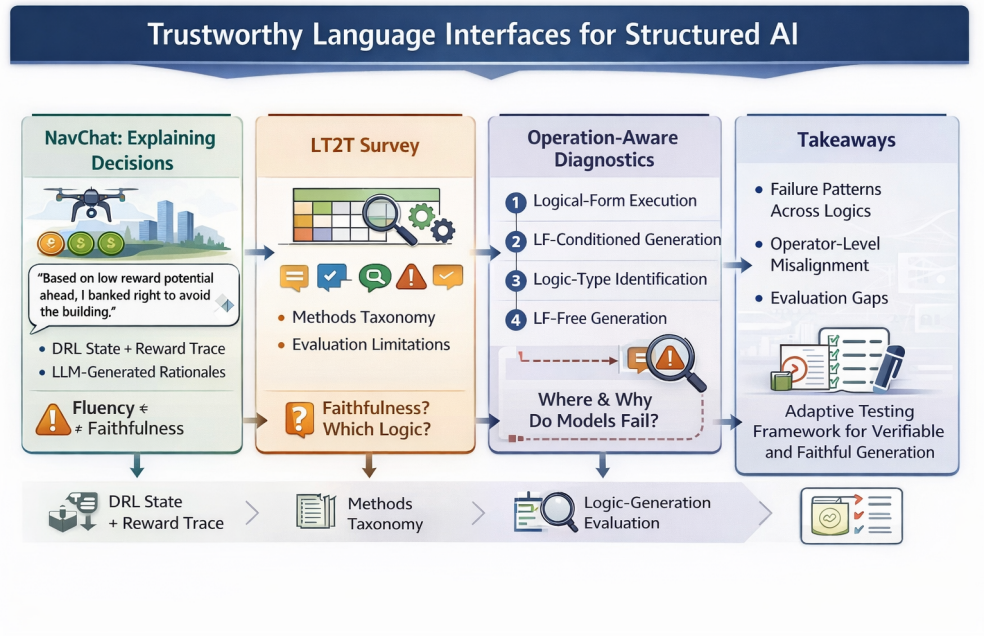


Figure 1.1: Overall structure of the dissertation (AI-generated).

namely structured episode traces, to condition an LLM that produces human-readable justifications for action sequences. We evaluate explanation quality and usefulness with a targeted human study and analyze common failure modes, highlighting numerical reasoning issues that limit trust in natural language rationales.

2. **Systematizing logical reasoning in table-to-text generation.** Although NavChat and LT2T arise in different application domains, both require models to translate structured evidence into natural language while preserving the underlying operations, comparisons, and derived values. Hence, to contextualize and systematize the logical reasoning failures observed in NavChat, we pivot to Logical Table-to-Text (LT2T), where tables provide a controllable sandbox for probing logical and numerical fidelity. We contribute a comprehensive survey of LT2T methods, models, datasets, and

core challenges, establishing a taxonomy of approaches and highlighting future research directions. This survey reviews how models are taught, guided, and evaluated for logical reasoning when they generate text from (semi)-structured data.

3. **Operation-aware diagnostic framework for LLM-based LT2T models.** Motivated by the limited diagnostic value of existing logical fidelity metrics, we introduce an operation-aware diagnostic framework for LLM-based LT2T methods. The framework decomposes performance into four competencies: (1) logical-form (LF) execution, (2) LF-conditioned generation, (3) logic-type identification, and (4) LF-free, type-controlled generation. By separating these competencies, we localize failures to specific operators, arguments, and grounding decisions, and we expose a recurring *meta-logical gap* in which models can generate plausible or even entailed statements while failing to preserve or recognize the intended underlying operation.

Together, these studies provide a unified perspective on trustworthy language interfaces for structured AI. NavChat demonstrates that explanation quality cannot be inferred from fluency alone; LT2T offers a principled environment for probing operation-level logical reasoning; and the operation-aware framework supplies the diagnostic granularity needed to understand *why* models fail, such as through incorrect row selection, incorrect calculations, or tie handling, rather than merely whether they fail.

Dissertation organization. Chapter 2 presents NavChat and its human evaluation. Chapter 3 provides the LT2T survey and method taxonomy. Chapter 4

introduces the operation-aware diagnostic framework and the empirical analysis across competencies and logic types. Finally, Chapter 5 summarizes contributions, discusses limitations, and Chapter 6 outlines future directions for explainable, verifiable language in structured AI.

Chapter 2

Natural Language Explanation for Deep Reinforcement Learning (DRL)-based Navigation

Autonomous navigation is a complex task because it requires analyzing extensive operational data to identify a safe and efficient path. Recently, many studies have used deep reinforcement learning (DRL) for navigation, and DRL has proven to be a powerful tool for such sequential decision-making problems. Despite being a promising tool, the opaque nature of the reasoning mechanisms of DRL methods makes them difficult to troubleshoot performance issues and fosters mistrust, thereby constraining their application in critical scenarios. This deficiency requires developing explanation methods to clarify the reasoning of DRL models for users. In this paper, we propose *NavChat*, an explanation framework designed to justify the trajectory suggested by a trained DRL model for the most efficient and safe path in an unknown environment. NavChat is a post-hoc approach that leverages a Large Language Model (LLM), and prompt engineering to generate

natural language explanations. The LLM provides natural language explanations that facilitate faster comprehension and greater clarity for non-expert users compared with visual explanations. We compare the explanations provided by NavChat for various prompts and then evaluate the accuracy and consistency of its explanations via a user study. This method facilitates understanding of the decisions made by the automated navigation model and communicates the reasoning process.

2.1 Introduction

Autonomous navigation is a complicated problem as it requires analyzing extensive operational data to identify an optimal and safe path. Failing to choose a secure route can have disastrous, potentially fatal consequences. However, recent advancements in machine learning techniques and their successful applications in processing complex problems have led to the development of autonomous navigation systems. Many of these studies employ DRL to address the navigation problem (Yan et al., 2020; Tong et al., 2021; Hu et al., 2021; Hodge et al., 2021; He et al., 2021). These works suggest that DRL can be a powerful tool for solving sequential decision-making problems. Without human involvement, DRL-based navigators can learn to choose a path from an initial point to a target point and avoid obstacles by interacting with an environment. These methods formulate the navigation problem as a sequential problem, wherein an agent selects each location of the path iteratively. Studies have demonstrated that DRL-based navigators can tackle the navigation problem by choosing the shortest and safest possible path without human interference (Yan et al., 2020; Tong et al., 2021; Hu et al., 2021; Hodge et al., 2021; He et al., 2021).

Although these DRL navigators can address routing problems well, there remains a challenging path ahead to integrate them into real-world systems. These autonomous systems have complex internal mechanisms that make their decisions difficult to interpret. Despite their potential, the opaque reasoning processes of DRL methods make them challenging to debug and contribute to a lack of trust, limiting their use in critical situations. This deficiency necessitates the development of explanation methods that clarify the reasoning process of DRL models for users.

Explainable DRL (XDRL) methods (Heuillet et al., 2021; Vouros, 2022; Hickling et al., 2023) have been proposed to explain an autonomous agent's decisions and to articulate the rationale behind its behavior. The sequential nature of DRL, the interactions between the agent and the environment, the evolving state of the environment, and the variety of reward signals make the design of XDRL methods challenging. Despite numerous XDRL methods in recent years, there is a noticeable lack of focus on explanation methods for DRL-based aerial navigation systems.

We propose NavChat, a post hoc explanation framework that generates natural-language *rationalizations* for the decisions made by a DRL UAV navigation policy. Here, *rationalizations* are human-readable justifications produced after the fact to help interpret why a complex model selected a particular action. In our setting, these rationalizations aim to explain and justify a DRL-generated trajectory that trades off efficiency and safety as it navigates an unknown environment.

NavChat leverages an LLM as a text generator conditioned on *structured decision evidence*; specifically, per-step state signals and reward-related information associated with each DRL action. In this work, we use GPT models (Brown et al., 2020) to translate these numerical traces into step-by-step explanations,

enabling scalable explanation generation without costly human annotation. However, NavChat also serves as a diagnostic lens: despite producing fluent and intuitive prose, LLM rationalizations can be *unfaithful*, meaning that they may sound plausible while misrepresenting the underlying state evidence, reward signals, or decision logic. Through prompt variations and a targeted user study, we evaluate the perceived quality and consistency of the generated explanations and report common reasoning failures, including numerical inconsistencies such as mis-handling negative deltas, and unsupported claims. These findings motivate the broader dissertation focus on how to evaluate and diagnose faithfulness in language generation from structured data.

2.2 Background

This section introduces the two main foundations of our work: reinforcement learning and the role of large language models in reasoning over structured evidence.

2.2.1 Reinforcement Learning

Reinforcement Learning (RL) is a framework within machine learning focused on sequential decision-making problems. Unlike supervised learning, where a model learns from labeled data, RL operates through evaluative feedback. In RL, an agent interacts with an environment to achieve a goal, receiving feedback in the form of rewards or penalties based on its actions. This feedback helps the agent learn an optimal *policy*—a strategy that dictates the best action to take in each state to maximize cumulative rewards over time. RL problems are often modeled as Markov Decision Processes (MDPs). An MDP is defined by: (1)

A set of *states* representing different situations or configurations the agent can encounter; (2) A set of *actions* that represents all possible moves the agent can make; (3) *Transition probabilities* that represent the likelihood of moving from one state to another, influenced by the agent's actions; (4) A *reward function*, which is a mechanism to evaluate the outcome of an action, providing positive or negative feedback to guide the learning process; (5) A *policy*, which is the strategy that the agent uses to decide its actions based on the current state; (6) A *value function* that quantifies the expected long-term sum of rewards from a given state, with future rewards typically discounted by a factor that weighs future rewards less than current rewards. An RL agent aims to learn an optimal policy that maximizes the expected cumulative reward, also known as the *return*.

Traditional RL relies on tabular methods to store value functions, which become infeasible for large state-action spaces. To fill this gap, *Deep Reinforcement Learning* (DRL) was developed by extending traditional RL through incorporating deep neural networks (DNNs) to handle the scalability issues faced by classic RL methods. While RL provides the theoretical framework and learning paradigms (e.g., policy optimization, value iteration), DRL replaced tables with DNNs to efficiently approximate value functions over vast state-action spaces. DRL scales to more complex tasks with a large number of states and actions, where traditional RL would struggle due to memory and computational constraints. A full explanation of the fundamental concepts of RL and DRL can be found in Sutton and Barto (2018).

2.2.2 Large Language Models as Reasoners

Language models (LMs) are models with the capability of understanding and generating natural language. These models are trained to predict the probability of the next words in a sequence.

Large Language Models (LLMs) are a class of deep neural network language models that typically contain tens or hundreds of billions of parameters (Chang et al., 2024) and are trained on huge amounts of text data, which enables them to perform a wide range of tasks, including text generation, translation, summarization, and answering questions.

One of the remarkable aspects of LLMs is their ability to exhibit apparent reasoning capabilities. "Reasoning" in the context of LLMs refers to the model's apparent capacity to make inferences, understand relationships, and solve problems based on the textual information it processes. This ability is crucial for applications that require understanding beyond mere pattern recognition, such as answering complex questions, solving logical puzzles, and engaging in meaningful dialogue.

One of the mechanisms that contribute to the reasoning abilities of LLMs is *prompt engineering* (Brown et al., 2020; Wei et al., 2022b), which is the practice of designing and optimizing input prompts to guide the behavior of LLMs. The design of input prompts can significantly influence the apparent reasoning capabilities of LLMs. By carefully crafting prompts, users can guide models to perform specific reasoning tasks more effectively.

Different kinds of prompt engineering, such as zero-shot, few-shot (Brown et al., 2020), and Chain-of-Thought (COT) (Wei et al., 2022b), have demonstrated that effective prompting can enable LLMs to provide suitable answers for

more complex reasoning tasks over text. These approaches show that instead of the costly approach of fine-tuning the model, the model can perform some degree of apparent reasoning. *Zero-shot* prompting uses only a task description or instruction as the prompt, without providing task-specific examples. In *Few-shot* prompting, the model is given a few input-output examples at inference time to demonstrate the desired task format and behavior. *Chain-of-Thought* prompting encourages the model to generate a series of intermediate natural-language reasoning steps before producing the final output. None of these prompting methods updates the model's parameters.

In addition to the prompting property, recent works have shown that LLMs have a promising ability to reason over unannotated data due to their training on vast amounts of text and patterns. In this work, we utilize these two properties of LLMs to generate natural language explanations for unannotated data extracted from a trained DRL navigation model.

2.3 Related Work

2.3.1 Explainable Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) models are often difficult to debug and interpret due to their dependency on numerous factors, such as the environment setup, reward design, observation representation, and even the specific training algorithms that shape the learned policy. Consequently, there is an increasing demand for Explainable Reinforcement Learning (XRL)—methods that make the decision-making process of RL agents more transparent and interpretable. Juozapaitis et al. (2019) introduced *reward decomposition*, where the total reward is

split into semantically meaningful components such as safety, speed, and efficiency, allowing actions to be explained in terms of trade-offs between these components. Madumal et al. (2020) advanced the field with a causal model of RL behavior, learning structural causal graphs that support contrastive "why/why-not" explanations and improve human understanding of agents' strategies. Together, these works mark key progress toward interpretable, human-aligned reinforcement learning systems. None of these works focus on explainability in DRL-based aerial navigation systems.

2.3.2 Explainable DRL-Based Aerial Navigation

Recent studies in explainable DRL for aerial navigation focus on making autonomous drone decisions more transparent and trustworthy. Works such as Çetin et al. (2023) and He et al. (2021) use visual interpretability tools like saliency maps, Grad-CAM, and feature attribution to highlight important image regions influencing navigation. Similarly, Thomas et al. (2021) employs self-attention visualization to show where the UAV model focuses during obstacle avoidance. While these methods help users see which sensory inputs guide actions, they do not explicitly verbalize the underlying decision logic. In contrast, NavChat complements these vision-based methods by providing a framework that converts internal state–action representations into natural-language explanations that mimic how a human would justify similar behavior.

2.4 Methodology

The goal is to provide justifications for the agents' actions based on environmental information, including state and reward data. As explained in Subsection 2.2.2,

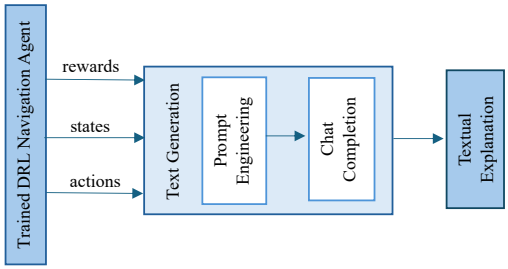


Figure 2.1: NavChat Framework

LLMs have shown a remarkable capability for apparent reasoning if the model is conditioned properly. Additionally, they can identify patterns and reason over unannotated data. Considering these two useful properties, we employ an LLM for both reasoning over our unlabeled data and generating textual explanations.

Prompt Engineering: We provide the state and reward information of the trained model in JSON format to the LLM and task it with explaining actions based on this information. This is not a trivial task, as the data is not annotated and the values are mostly numerical, requiring numerical reasoning for understanding and interpretation.

To promote the reasoning ability of the LLM, we use different prompting, including zero-shot, numeric, and COT. These prompts have a common part that sets the behavior and objectives for the model; it tells the model how it should think and respond before any task-specific input is given. In our case, the system message makes the model act as an evaluator for an autonomous agent's decisions. Also, it explains the scenario for the agent. Figure 2.2 shows this part of the prompt called *Role*.

Zero-Shot: The zero-shot prompt serves as the baseline configuration. In this setting, the LLM is presented with the agent's state–reward information and instructed to evaluate an action without seeing any prior examples. The

Role: You are responsible for evaluating each action taken by an agent based on the given information in each timestep.

The agent tries to find the shortest and safest path from a start position to a target. There are obstacles on the way to the target, so the agent should consider the safety of the path and avoid obstacles.

The impact of the action is provided as information in each timestep. You are given the information of two sequential timesteps; consider the first one as the previous timestep and the second one as the current timestep. For instance, you have information for timestep zero and timestep one. In this case, the information for timestep zero is considered the previous timestep, while the information for timestep one is considered the current timestep. You are required to evaluate the action in the current timestep, which is timestep one in this example.

The *objective relative distance* shows the difference between the previous distance and the current distance from the objective.

The *unvisited penalty* is the distance from the current position to the objective.

Figure 2.2: Common part of all prompts (Role).

Task:

- Consider the information of "timestep 0" as the initial condition. When the timestep is zero, the distance objective is the initial distance to the target position.
- You have the information for two consecutive timesteps; consider the first one as the previous timestep and the second one as the current timestep.
- Evaluate only the action of the current timestep.
- Compare the values of the current timestep with the previous one.
- Find some rules that can be used to justify your evaluation.
- Then, decide whether the action was very good, good, neutral, bad, or very bad, and explain your decision.
- You can also explain based on the rules that you find.

Output:

- The action is very good, good, neutral, bad, or very bad because...

Figure 2.3: Zero-shot prompt.

When comparing two negative values, it is opposite comparing the positive values. For example, -0.55 is less than -0.42, or -0.42 is greater than -0.55.

Figure 2.4: One-shot prompt.

Compare the values of the current timestep with the previous one such as distance objective, yaw objective, and obstacle and wall penalties. consider following examples:

- distance objective of the previous timestep is 0.5879, and distance objective of the current timestep is 0.1547. Does the distance reduce or increase? Is the reduction substantial or slight?
  - ✓ Thought process: 0.5879-0.1547 = 0.4332, so the distance to the objective was reduced by 0.4332.

  compare the reduction with the threshold: 0.4332 > 0.04
  - ✓ Answer: the distance to the objective was reduced because the reduction is positive value. The reduction is substantial because the amount of the reduction is greater than 0.04. The action is effective due to a substantial decrease.
- distance objective of the previous timestep is 0.1254, and distance objective of the current timestep is 0.1687. Does the distance reduce or increase?
  - ✓ Thought process: 0.1254 - 0.1687 = -0.0433, that is the distance to the objective was increased by 0.0433.
  - ✓ Answer: the distance to the objective was increased because the reduction is positive value. The action is ineffective.
- distance objective of the previous timestep is 0.1254, and distance objective of the current timestep is 0.1237. Is the reduction substantial or slight?
  - ✓ Thought process: 0.1254 - 0.1237 = 0.002, compare the reduction with the threshold: 0.002 < 0.04
  - ✓ Answer: slight because the amount of reduction is less than 0.04. Although the reduction is positive, the
  - ✓ action is not effective due to a slight reduction.

Figure 2.5: Sample of Chain of Thought prompt

prompt simply defines the task and output format, guiding the model to compare numerical values such as distance to target, penalties, and yaw objective across two consecutive timesteps. As shown in Figure 2.3, the LLM is directed to determine whether the action in the current timestep is very good, good, neutral, bad, or very bad, and to justify this decision based on its internal reasoning. This setup allows us to assess the LLM's inherent ability to perform numerical reasoning and relational comparison without any external guidance. However, in practice, the model often struggles to interpret negative deltas and to formulate consistent evaluative rules, which motivates the subsequent one-shot refinement.

One-Shot: In the one-shot configuration, we extend the zero-shot setup by including a single worked example illustrating the desired reasoning process. This example demonstrates how to compare numerical differences—especially handling negative values, which the model frequently misinterprets in zero-shot mode. By presenting a reference case that explicitly explains how to compute and interpret

these differences, the one-shot prompt provides a conceptual anchor that stabilizes the model's logic.

Chain of Thought: The Chain-of-Thought prompt (see Figure 2.5) introduces explicit step-by-step reasoning examples that decompose each evaluation into intermediate mathematical and logical steps. For instance, the prompt shows how to calculate the difference between the previous and current objective distances, compare it against a threshold, and then interpret the reduction as substantial, slight, or negative. Each example includes a "Thought process" section, which models the arithmetic reasoning the LLM should emulate before producing the final explanation. This scaffolding teaches the model to structure its reasoning: first performing numerical computation, then drawing a qualitative conclusion, and finally articulating the rationale in natural language.

2.5 Experiments and Results

2.5.1 DRL Navigation system and environment

In our experiments, the DRL navigation agent is trained using Proximal Policy Optimization (PPO) (Schulman et al., 2017) in a 2D simulation environment where there are five obstacles numbered from zero to four, one target, and a simulated uncrewed aerial vehicle (UAV). At the beginning of each episode, the position of each of these components changes randomly; therefore, the agent learns to find paths for various cases. In all scenarios, the obstacles are stationary and only the UAV moves. The state of the environment consists of 12 features: the relative distance of the UAV to the target, the relative yaw of the UAV to the target, the relative distance of the UAV to each obstacle (5 obstacles), and

the relative yaw of the UAV to each obstacle (5 obstacles). The action space is continuous, and each action includes two parts: (1) the relative distance of the UAV to the target, and (2) the relative yaw of the UAV to the target. Obstacles are located randomly, and there is an unsafe zone around them; if the UAV enters any of these zones, it will be penalized. The agent receives rewards based on the distance and safety, i.e., if it manages to stay away from the unsafe zones and gets closer to the target, it will be rewarded; otherwise, it will be penalized if it enters an unsafe zone or moves further from the target. The episode ends when either the UAV reaches the target within an acceptance radius or crashes into an obstacle or boundary wall around the navigation environment.

2.5.2 Text Generation Results

In evaluating the generated explanations by GPT, we focus on two main objectives:

1. **Objective 1: Evaluate the numerical reasoning, resistance to hallucinations, and faithfulness of the generated text.** *Numerical reasoning* refers to the model's ability to understand and manipulate numbers. In our case, the model should be able to do simple arithmetic operations like subtraction and comparison between numbers, to explain and justify actions. Thus, we analyze the correctness of the numerical reasoning. *Resistance to hallucinations* assesses the accuracy of the explanation and its conformance to factual information available or inferable (i.e., the agent should not invent falsehoods that are presented as factual), and *faithfulness* determines whether the generated text accurately reflects the input data, such as the correctness of the values of the features.

2. **Objective 2: Assess the usefulness of the generated explanations for users.** We evaluate the extent to which the generated explanations enhance users' understanding of the agent's behavior.

We use human evaluation to analyze both goals. In the following, we describe the methodology and results of each objective.

Objective 1

As described in Objective 1, this section evaluates the correctness of the reasoning, resistance to hallucination, and faithfulness of the generated explanations. For this task, we used OpenAI ChatGPT Completion API (OpenAI, 2020) for text generation, and we tested three different models: GPT-3.5-Turbo, GPT-4 Turbo, and GPT-4o. We set the temperature parameter to 0.2 to get deterministic outcomes.

In this experiment, we provided the GPT model with an episode data file containing changes in state features and reward values across that episode. Then, we tasked the model with evaluating the actions selected by the DRL navigation model of the episode. The GPT model was instructed to compare the values of the features for two sequential time steps and categorize each action as very good, good, bad, or very bad based on three criteria: distance, safety, and alignment. Besides evaluating the actions, the model should also explain its reasoning with respect to the changes in the environment. For instance, the action in time step x is very good because the changes in the distance show that the UAV gets closer to the target, it is safe as there is no penalty, and it is aligned as it moves towards the target.

We used three different prompting methods: zero-shot, numeric-prompt, and

Table 2.1: The accuracy of different models in evaluating the actions using different prompting methods.

Action Evaluation			
Model \ Prompting	Zero-shot	Numeric	COT
GPT-3.5-Turbo	47%	66%	88%
GPT-4 Turbo	89%	89%	94%
GPT-4o	100%	94%	100%

chain of thought (COT). Zero-shot prompting simply narrated the description of the task and explained how to categorize an action. We noticed some numerical reasoning errors in the zero-shot results. To address this, we added an explanation about the changes in the distance and how to interpret this change; we call this prompt *numeric-prompt*. Finally, in order to enhance the reasoning ability of the model, we utilized COT prompting. COT prompting provided detailed examples of some arithmetic operations. For instance, it contained the thought process of analyzing, comparing, and interpreting the changes in the values of the features, such as relative yaw to objective on two consecutive time steps.

We selected three different episodes where the agent found a path from the start point to the target successfully, although the paths were not optimal. For each episode and each model, we conducted five trials and considered the majority response as the final evaluation. For example, if out of five repetitions, the LLM responded *very good* three times and *good* twice, the response was determined to be *very good*. The final result is the average evaluation accuracy across the three episodes. The results of accuracy in evaluating the actions are detailed in Table 2.1.

Among the three models, GPT-4o generated the most accurate evaluations and justifications across all metrics. In the three episodes evaluated, it encoun-

```
{
  "timestep": 3,
  "state": {
    "relative_yaw_UAV_to_target": -0.1603,
    "relative_distance_UAV_to_target": 0.1196,
    "relative_yaw_UAV_to_obs0": 0.1391,
    "relative_yaw_UAV_to_obs1": 0.5909,
    "relative_yaw_UAV_to_obs2": 0.4475,
    "relative_yaw_UAV_to_obs3": 1.3874,
    "relative_yaw_UAV_to_obs4": -0.4209,
    "relative_distance_UAV_to_obs0": 0.8293,
    "relative_distance_UAV_to_obs1": 1.0319,
    "relative_distance_UAV_to_obs2": 0.4794,
    "relative_distance_UAV_to_obs3": 0.663,
    "relative_distance_UAV_to_obs4": 0.3681
  },
  "reward_components": {

    "obstacle_penalty": 0
    "wall_penalty": 0.0
  }
}
```


(a) State information of time step3.


```
{
  "timestep": 4,
  "state": {
    "relative_yaw_UAV_to_target": -1.743,
    "relative_distance_UAV_to_target": 0.5436,
    "relative_yaw_UAV_to_obs0": -0.582,
    "relative_yaw_UAV_to_obs1": 0.0903,
    "relative_yaw_UAV_to_obs2": -0.9499,
    "relative_yaw_UAV_to_obs3": 2.1586,
    "relative_yaw_UAV_to_obs4": -1.3858,
    "relative_distance_UAV_to_obs0": 0.7304,
    "relative_distance_UAV_to_obs1": 0.6484,
    "relative_distance_UAV_to_obs2": 0.3799,
    "relative_distance_UAV_to_obs3": 0.1628,
    "relative_distance_UAV_to_obs4": 0.6784
  },
  "reward_components": {

    "obstacle_penalty": -8.37
    "wall_penalty": 0.0
  }
}
```


(b) State information of time step4.

Figure 2.6: State information of two consecutive time steps.

tered only one numerical reasoning issue. Additionally, all explanations were factual, with no hallucinations or fidelity issues. GPT-4 Turbo ranked second, with a few numerical reasoning errors but no hallucinations or fidelity issues. GPT-3.5-Turbo created the least accurate evaluation in terms of both classifying the actions based on the provided category and justification. The primary issues included weak numerical reasoning, which improved somewhat with prompt engineering but remained less accurate than the GPT-4 models, failure to follow prompt instructions, and occasional hallucinations and incorrect explanations. Additionally, GPT-3.5-Turbo sometimes reported incorrect values from the input file, affecting its faithfulness.

An example is given in Figure 2.6, which depicts data of two time steps of one of the selected episodes. Figures 2.7 and 2.8 illustrate the evaluations and explanations for time step 4 based on data in Figure 2.6, generated by GPT-3.5-Turbo and GPT-4o, respectively. A close comparison between the results shows that GPT-3.5-Turbo categorized the action incorrectly due to numerical reasoning and fidelity issues, highlighted in red. In contrast, GPT-4o correctly classified the action with valid explanations marked in green.

The action is very good because:
1. Effective: the agent successfully reduced the distance to the target by 0.4240 (0.5436 - 0.1196), which is more than 0.04.
2. Safe: there is no penalty incurred for obstacles or walls.
3. Aligned: The action is also aligned as the yaw objectives have changed towards the target.

Figure 2.7: Evaluation and justification generated by GPT-3.5.

The action is very bad because:
1. Ineffective: The distance to the objective increased from 0.1196 to 0.5436. This is a significant increase, indicating that the agent moved further away from the target.
2. Unsafe: The obstacle penalty increased to -8.37, indicating that the agent moved closer to obstacles or encountered them.
3. Unaligned: The yaw objective changed from -0.1603 to -1.743, which is a substantial deviation, indicating that the agent's orientation is less aligned with the target.

Figure 2.8: Evaluation and justification generated by GPT-4o.

Regarding the different prompts, we observed that the results generated by GPT-3.5 improved substantially as we provided more detailed prompts and instructed it on how to think and analyze. Although the GPT-4 models already produced better results, they can still benefit from prompt engineering.

Because the GPT-4o model produced better explanations compared to GPT-3, we will continue the next experiment with the former.

Operation-Aware Diagnosis of Explanations

This section reports the common logical errors that we observe. A qualitative inspection of NavChat's chain-of-thought outputs reveals that many failures are not merely stylistic or semantic, but arise from errors in executing a small set of *atomic operations*. In particular, the prompt requires the model to reason over (1) signed subtraction for distance reduction, (2) threshold comparison against 0.08, (3) absolute-value transformation for yaw, and (4) comparison of these absolute values against the threshold 1 to determine whether a directional shift is acceptable or drastic. The prompt also explicitly warns that negative values must be interpreted carefully and that yaw should be handled via absolute values

rather than direct subtraction.

Despite these explicit instructions, the outputs exhibit recurring operation-level failures. First, the model repeatedly makes **absolute-value and threshold comparison errors**. For example, one output computes $|-1.2589| = 1.2589$ and $|-0.8232| = 0.8232$, but then incorrectly states that *both* values are greater than 1 and therefore the shift is acceptable. Similar errors appear elsewhere, such as claiming that both $|-0.9035|$ and $|-0.9556|$ are greater than 1, even though both are in fact less than 1.

Second, NavChat shows **distance-thresholding errors** around the effectiveness rule. The prompt defines an action as effective only when the reduction in distance is positive and exceeds 0.04. However, the model sometimes treats slight reductions as substantial. For instance, one output states that $0.2491 - 0.2372 = 0.0119$ is a slight reduction and therefore ineffective, which is correct, while other outputs on nearby cases misclassify similarly small reductions or produce inconsistent verbal judgments relative to the same threshold. This suggests brittleness even on the simplest arithmetic comparison step.

Third, the model exhibits **sign-sensitive reasoning failures** for yaw. In several cases, it inconsistently handles transitions involving negative values, especially when one magnitude is above 1, and the other is below 1. For example, the transition from -1.5488 to -0.5103 is sometimes treated as aligned and acceptable, even though another output correctly recognizes that the change crosses the $|y| = 1$ boundary and should therefore be considered drastic. This inconsistency is notable because the prompt explicitly emphasizes careful handling of negative values.

Fourth, and most strikingly, the model frequently fails in **rule composition from attributes to final label**. The prompt provides an explicit mapping from

Factor	Decision Rule	Label
Effectiveness	Compute $\Delta_d = d_{\text{prev}} - d_{\text{curr}}$. substantial reduction: if $\Delta_d > 0.04$. slight reduction: if $0 < \Delta_d \leq 0.04$.	Effective, if $\Delta_d > 0.04$ Ineffective, otherwise
Alignment	Compare $\lvert y_{\text{prev}} \rvert$ and $\lvert y_{\text{curr}} \rvert$ to 1. acceptable shift: if both are > 1, or both are < 1. drastic shift: if one is > 1 and the other is < 1.	Aligned, acceptable shift Unaligned, a drastic shift
Safety	Check obstacle and wall penalties in the current timestep. safe: if both penalties are 0. unsafe: if either penalty is nonzero.	Safe, if **No** penalty Unsafe, otherwise

Table 2.2: Rules used to determine effectiveness, alignment, and safety before assigning the final action label.

Effective	Aligned	Safe	Label
✓	✓	✓	Very Good
✓	×	✓	Good
✓	✓	×	Good
✓	×	×	Neutral
×	✓	✓	Bad
×	✓	×	Bad
×	×	✓	Bad
×	×	×	Very Bad

Table 2.3: NavChat action-category mapping.

the three binary properties to the final action category as depicted in Table 2.3. Yet many outputs contradict this mapping. In one case, the model states that the action is *safe and aligned but ineffective*, which should map to *bad*, and indeed does so correctly. But in other cases, it states that the action is effective, safe, and aligned and then outputs *neutral*, or describes the action as ineffective, safe, and aligned yet concludes *very bad*. These are not vague hallucinations; they are failures to compose already-derived intermediate judgments into the correct final decision.

Taken together, these examples suggest that fluent explanations can mask brittle operation execution. Even when the reasoning scaffold explicitly specifies the needed operators and thresholds, the model may still fail on signed arithmetic, absolute-value comparison, thresholding, or final rule composition. We therefore view NavChat as an early motivating case study for the broader dissertation question: whether current models can perform reliable *operator-level reasoning* over structured evidence, and how such failures can be measured and diagnosed in a fine-grained, explainable manner.

Objective 2

We investigate the usefulness of the generated explanations for users through a user study. The primary objective of our user study is to evaluate the extent to which NavChat helps users understand the decisions made by the DRL agent.

The user study was conducted over one week, and all participants were directly contacted by the author. We used an online questionnaire through Qualtrics, via a university account. In the course of 7 days, we collected 10 completed questionnaires. The participants were mostly PhD students and researchers, either in academia or industry, across the United States. In the following, we describe the user study in detail.

We started the survey by describing the goal of the study, outlining the steps for participants, and asking for their consent. After the introduction, we asked the participants to rate their knowledge of machine learning, reinforcement learning, and AI chatbots using a 5-point scale: not at all familiar, slightly familiar, somewhat familiar, moderately familiar, and extremely familiar. To familiarize participants with the problem, we briefly explained how reinforcement learning works, defined some concepts related to the problem, and then explained the

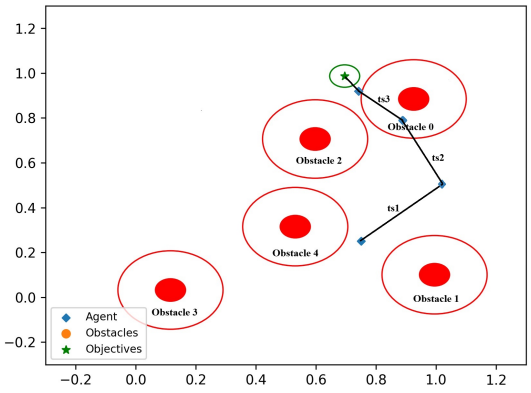


Figure 2.9: The trajectory of the scenario in the user study

Table 2.4: Questions used in the user study

No	Question
1	At time step 02, did the agent select a good action?
2	Provide a list of time steps that the agent could select a safer action.
3	In how many of the actions does the UAV get closer to the target?
4	How many of the actions are not aligned?
5	Could the agent reach the target faster?
6	What is your evaluation of the selected path for the entire episode?
7	Provide a list of time steps that the agent could select a better action.

scenario. The scenario contained information about the environment, actions, and the variables that change the state of the environment. Figure 2.9 shows the trajectory of the scenario used in the user study. After these introductions, participants were asked to complete three tasks.

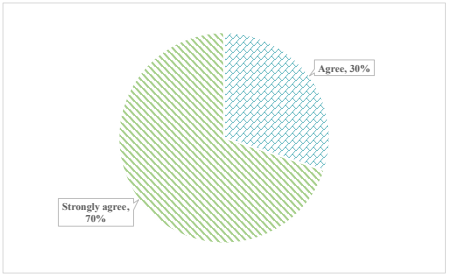
In Task 1, participants were required to answer seven questions listed in Table 2.4 using the information given in a JSON file, which is comprised of changes in the states for the entire episode of the scenario. The same questions and data were provided to NavChat for generating textual explanations. The main goal of this step was to acquaint participants with the scenario analyzed by NavChat and highlight the difficulties in justifying the DRL agent's decisions by observing the raw data extracted from DRL states.

In Task 2, participants were shown the explanations generated by NavChat for the same questions they answered in Task 1. Participants were then asked to read and rate their degree of agreement with NavChat's explanations for each question using a 5-point scale: strongly agree, somewhat agree, neither agree nor disagree, somewhat disagree, and strongly disagree. In addition to rating the justifications of the seven questions, participants also rated their overall agreement with the explanations generated by NavChat using a 7-point scale: strongly agree, agree, somewhat agree, neither agree nor disagree, somewhat disagree, disagree, and strongly disagree. This step allowed participants to evaluate NavChat's reasoning ability by comparing the justifications with their own thinking and assessing the value of the explanations.

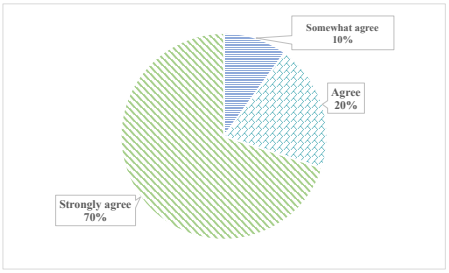
In Task 3, in addition to the data file, questions, and answers from NavChat, participants were presented with an animation similar to Figure 2.9, which shows the movement of the UAV, as well as the static positions of the target and the obstacles with their danger perimeter zones. Building on what was required in Task 2, Task 3 included an additional question asking participants to rate the usefulness of different explanation methods. Participants were asked to rank the usefulness of the data file only, NavChat's explanations plus data file, NavChat's explanations only, animation only, and animation plus NavChat's explanations. The ranking is between 1 and 5, where 1 is the lowest and 5 is the highest rank.

By incorporating the animation, we aim to compare the effectiveness of textual explanations against visual representations of the problem. This will help us determine whether participants find the textual explanations equally useful when accompanied by an animation, or if they perceive the textual explanations to be less helpful in the presence of visual aids.

Figures 2.10a and 2.10b compare the overall degree of the usefulness of the



(a) The overall usefulness of NavChat's explanations for all questions (Task 2).



(b) The overall usefulness of NavChat's explanations for all questions (Task 3).

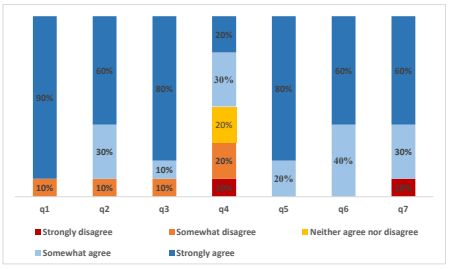
Figure 2.10: Plots demonstrating the overall agreements in two tasks.

explanations for all questions in Task 2 and Task 3, respectively. In Task 2, without having the visualization of changes in the environment, all participants rated the usefulness of the generated explanations 100%, where 70% strongly agreed, and 30% agreed. This trend remains almost the same after adding the visualization; still, participants agree 100% that the explanations are helpful.

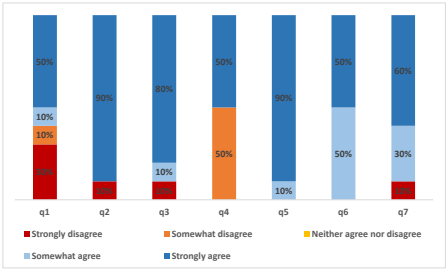
Figures 2.11a and 2.11b depict the participants' satisfaction with individual questions listed in Table 2.4. A comparison between the two charts shows that participants mostly change their minds about Questions 1 and 4 after adding the animation.

Question 1 is about evaluating the action in time step 2, as depicted in Figure 2.9, in which Action 2 reduced the distance to the target, was partially not aligned, and it was in a danger area. We believe adding the animation helped the participants see this clearly, leading those who had not imagined it correctly in Task 2 to change their minds in Task 3.

Question 4 is a tricky question, as different participants might interpret alignment differently, and it seems they had a hard time deciding about NavChat's explanation in Task 2. Adding the visualization in Task 3 seems to have helped them better assess the usefulness of the explanations provided by NavChat.



(a) The extent of agreement with NavChat's explanation for each question (Task 2).



(b) The extent of agreement with NavChat's explanation for each question (Task 3).

Figure 2.11: Plots demonstrating the extent of the agreements with NavChat's explanation for each question in two tasks.

Finally, the ranking of the different methods is demonstrated in Figure 2.12. Participants ranked the animation plus NavChat's explanations as the most useful method and the data file alone as the least useful method in understanding the DRL agent's decisions.

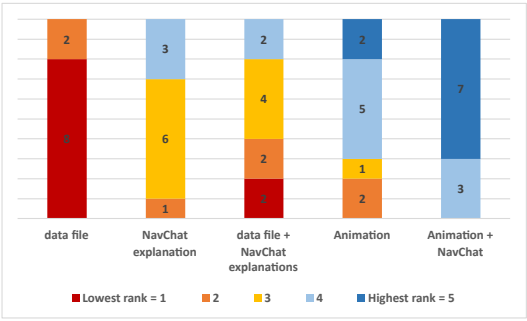


Figure 2.12: Ranking of various methods of explanations.

2.6 Conclusions and Future Work

We introduced *NavChat*, a post-hoc explanation framework that aims to improve human understanding of decisions made by a DRL-based UAV navigator. NavChat leverages an LLM-powered chat interface to translate structured naviga-

tion traces, state, and reward information into natural language rationalizations of the agent's actions. Results from our user study indicate that these explanations can improve users' perceived transparency and understandability of an otherwise opaque decision-making system.

While NavChat demonstrates the promise of LLM-assisted rationalization, it also highlights a central limitation: fluent explanations are not necessarily *faithful* to the structured evidence that underlies the agent's decision process. Our immediate future work will therefore extend NavChat along directions that directly address the broader research question, *Do current models possess reliable operation-level reasoning over structured evidence, and how can we measure and diagnose that reliability?*

The current method mostly relies on OpenAI's LLMs. In future work, we will investigate other LLMs, some of which are reported to have more powerful reasoning ability.

Chapter 3

Logical Table-to-Text Generation: Challenges, Methods, and Reasoning

Logical Table-to-Text (LT2T) generation requires models to both verbalize tabular data and reason over it—performing comparisons, aggregations, and causal inference. While many generation tasks struggle with similar analytical demands, LT2T provides a structured perspective on reasoning capabilities in natural language generation. This survey uses LT2T as a lens to focus on reasoning in data-to-text tasks. By focusing narrowly on LT2T, we present a deep taxonomy of methods that inject, structure, or verify reasoning steps, allowing a level of technical granularity missing in broader surveys. We review representative models and evaluation metrics, and highlight how LT2T techniques transfer to general generation challenges involving logic, numeracy, and faithfulness. Our goal is to distill lessons from LT2T that apply more widely, while also guiding future research in table-based reasoning.

Statistics of Three Countries in Middle East			
Country	**Area (km^2)**	**Population**	**Density (per km^2)**
Israel	21,937	10,100,000	460
Iraq	438,317	45,521,000	104
Iran	1,648,195	87,500,000	53
Surface-level Generation			
Sentence: Israel has an area of 21,937 km^2 and on average 460 persons live in each km^2.			
Logical-level Generation			
Sentence: Israel has the smallest area among the three countries with the highest population per square kilometer.			

Figure 3.1: Comparing surface and logical-level outputs.

3.1 Introduction

Tabular data is pervasive in domains such as finance, science, and sports, but is often obscure to non-experts. *Table-to-Text* (T2T) models bridge this gap by transforming structured tables into readable summaries. While early neural T2T models improved fluency, they struggled to meet users' analytical expectations, often producing surface-level descriptions that merely repeat table content.

Logical Table-to-Text (LT2T) raises the bar by requiring reasoning-based generation. Instead of stating individual facts, LT2T models must infer logical relationships. Figure 3.1 contrasts shallow and logical outputs to illustrate this difference. Generating logical-level output introduces challenges such as logical fidelity, numerical accuracy, and controllable content selection, which also affect many other natural language generation (NLG) tasks.

We chose LT2T to survey as it offers a compact, well-defined testbed for the broader NLP goal of trustworthy text generation with reasoning because the input structure is explicit and every valid operation is enumerable. At the same time, limiting the scope to LT2T allows us to offer a deep, actionable taxonomy

of reasoning strategies that would be too shallow or generic in a broader survey. We categorize methods by how they address issues to enhance reasoning. This is especially useful for practitioners entering the field or designing systems that require interpretable, faithful reasoning.

This survey, therefore, treats LT2T as a unique frame of reference for reasoning -centric generation research. Following background on LT2T and reasoning (Section 3.2), we review the primary datasets developed for LT2T (Section 3.3) and models frequently used in LT2T (Section 3.4). We outline the key challenges that arise in this task (Section 3.5), and organize existing methods into a unifying taxonomy (Section 3.6). Our taxonomy links challenges to methods, enabling clearer comparisons and identification of underexplored directions. We then compare these methods (Section 3.7), show datasets and evaluation metrics used by surveyed papers (Section 3.8), discuss how to select them (Section 3.9), then present future directions (Section 3.10), and outline conclusions (Section 3.11). We hope this survey serves both as a roadmap for LT2T and as a blueprint for reasoning-aware generation more generally.

3.2 Background

This section briefly defines the Logical Table-to-Text task and discusses the types of reasoning used.

3.2.1 Logical Table-to-Text

The *Logical Table-to-Text* (LT2T) task is to learn a mapping from a table to an articulate natural language sentence that can be derived from the input table. Formally, the task can be defined as follows (Chen et al., 2020a):

Given a table T denoted as $T = \{ T_{i,j} \mid 1 \leq i \leq R_T,\ 1 \leq j \leq C_T \}$, where R_T and C_T are the number of rows and columns, respectively, and each cell entry $T_{i,j}$ may contain a word, a number, a phrase, or even an entire sentence; and reference sentence(s) of the form $W = (w_1, w_2, \ldots, w_n)$ composed of words w_i; the objective is to train a model $P(W \mid T)$ that generates a hypothesis $\hat{W}$ that is both (i) *fluent* and (ii) *logically entailed* by the information in T.

3.2.2 Reasoning

Reasoning involves analyzing facts to infer new insights, draw conclusions, or make decisions by identifying patterns, comparisons, or causal relationships within data. In LT2T, reasoning enables the generation of logically correct statements that extend beyond surface-level information. This survey focuses specifically on two types of reasoning:

Logical: *Logical reasoning* is inferring analytical relationships such as comparisons, superlatives, or causal connections. For example, given statements "A is taller than B" and "B is taller than C," logical reasoning can conclude that "A is taller than C."

Numerical: *Numerical reasoning* refers to making correct inferences from numerical data, including arithmetic operations, magnitude comparison, and numeric aggregation. For example, determining the tallest individual is A given specific heights {A: 180 cm, B: 170 cm, C: 160 cm}.

Although numerical reasoning is a subset of logical reasoning, it is considered separately here due to the specialized challenges it presents for LT2T.

Dataset	Tables	Cell	Num.	Pairs	Vocab	Text	Domain	Source	Reasoning	Schema	Cont.	Struct.
LogicNLG	7.3K	91	35	37.0K	122K	14	Open (Wiki)	Annot.	Rich (Logical)	Unlimited	No	Flat
Logic2Text	5.5k	64*	22*	10.7k	14k	16.7	Open (Wiki)	Annot.	Rich (Logical)	Unlimited	No	Flat
NumericNLG	1.3K	35*	31*	1.3K	19.6K	94	Scientific	Hybrid	Rich (Numerical)	Unlimited	No	Flat
SciGen	—	53	34	1.3K	11K	116	Scientific	Annot.	Rich (Numerical)	Unlimited	No	Flat
ContLOG	5.5K	64*	22*	10.7K	14K*	16.7*	Open	Annot.	Rich (Logical)	Unlimited	Yes	Flat
LoTNLG	862	84*	25*	4.3K	6.3K*	14*	Open	Annot.	Rich (Logical)	Unlimited	Yes	Flat
HiTab	3597	190*	116*	10.6K	8.7K*	17.32*	Open	Annot.	Rich (Numerical)	Unlimited	Yes	Hier.

Table 3.1: Logical Table-to-Text Datasets.

3.3 Datasets

Training a generative model for LT2T requires a specific dataset, which typically consists of pairs of tables and narratives that describe table contents. Numerous datasets exist for generating descriptive text from tables, such as WikiBio (Lebret et al., 2016). However, our focus is on datasets specifically designed to challenge and evaluate the logical/numerical reasoning capabilities of LT2T techniques. Table 3.1 summarizes LT2T datasets. Here, *Tables* denotes the number of tables, *Cell* the average number of cells per table, *Num.* the average number of numeric cells per table, and *Pairs* the number of annotated table–text pairs. *Vocab* and *Text* indicate vocabulary size and average description length, respectively. *Source* describes the data origin, *Schema* indicates whether the schema is focused on a specific topic or open-ended, *Cont.* denotes whether the dataset guides what content to verbalize, and *Struct.* specifies whether the table structure is flat or hierarchical.

In the following, we first discuss what is currently considered in the design of existing LT2T datasets and then what is missing.

LogicNLG (Chen et al., 2020a) increases logical difficulty through open-domain, unconstrained schemas and diverse logical operations such as superlatives and comparisons. The dataset's tables span multiple domains, though the distri-

bution is heavily skewed toward sports: approximately 35% of tables concern teams/players and 25% concern competitions, followed by entertainment and politics at roughly 15% each, with celebrity and science domains appearing far less frequently.

Logic2Text (Chen et al., 2020b) pairs tables with logical forms (LFs); it also *quantifies structural reasoning complexity* using LF graph size, including number of nodes and function nodes. Each logical form contains on average nine nodes, including about three function nodes. The dataset includes seven logic types: Count (2.0k) and majority (1.8k) are the most common, followed by unique (1.6k), superlative and aggregation (1.4k each). Ordinal and comparative (1.2 to 3k each) are less frequent. This distribution shows that Logic2Text emphasizes summarization and highlighting prominent rows, while explicit comparison and ranking operations are present but occur less often.

NumericNLG (Suadaa et al., 2021) and *SciGen* (Moosavi et al., 2021) target numerical reasoning by emphasizing arithmetic operations; *SciGen* further introduces three splits, with the hardest split containing longer, more analytical statements.

ContLOG (Liu et al., 2022) is a controllable dataset built on Logic2Text that replaces LFs with highlighted evidence cells to guide logical content selection; it also provides a pre-training subset.

LoTNLG (Zhao et al., 2023d) is an evaluation-only benchmark designed for zero-shot/prompted LLM testing, conditioning generation on nine reasoning types to probe type-specific difficulty.

All of the above operate on flat tables. *HiTab* (Cheng et al., 2022) makes the task harder by introducing hierarchical (multi-level header) tables and shows that schema hierarchy stresses alignment and reasoning.

Gaps and Directions. Despite these contributions, current LT2T datasets leave room for growth in at least three ways: (1) *Schema realism:* include hierarchical headers and *multi-table joins* to reflect real reports and dashboards. (2) *Operation coverage:* cover more operations such as temporal (trends, deltas, rolling statistics). (3) *Complexity metadata:* release per-dataset distributions such as operation types, domains, and numeric density.

3.4 Models

We categorize the models commonly used in LT2T works into three groups.

Traditional Encoder-Decoder Models: Early works train encoder-decoders such as Gated Recurrent Neural Network (GRU) (Chung et al., 2014), Long Short Term memory (LSTM) (Hochreiter and Schmidhuber, 1997), or vanilla Transformers (Vaswani et al., 2017) from scratch. These models are usually augmented with structural add-ons, such as copy mechanisms (Gu et al., 2016) or pointer networks (See et al., 2017), gating (Liu et al., 2018), or attention mechanisms (Luong et al., 2015) to ensure numbers and entity names are reproduced faithfully. Methods that use these models are presented in Table 3.2.

Method	Base Model	Attention	Gating	Copy	Training Method
OpAtt (Nie et al., 2018)	GRU	Yes	Yes	Yes	MLE
CP-DU (Gong et al., 2020)	LSTM/Trans.	Yes	Yes	Yes	RL
Field-Gating (Chen et al., 2020a)	LSTM/Trans.	Yes	Yes	Yes	MLE
Field-Infusing (Chen et al., 2020a)	LSTM/Trans.	Yes	No	Yes	MLE
DCVED (Cheng et al., 2021)	Trans.	Yes	No	No	CI

Table 3.2: Traditional encoder-decoders trained from scratch for LT2T. MLE = Maximum Likelihood Estimation, RL = Reinforcement Learning, CI = Causal Intervention, Trans = Transformer

Pre-trained Language Models (PLMs): Subsequent studies fine-tune generic

Method	Base Model	Param. Size	Training Method
BERT-TabGen (Chen et al., 2020a)	BERT-*small*/*large*	140/340M	MLE
GPT-TabGen (Chen et al., 2020a)	GPT2-*small*/*medium*	117/345M	MLE; Adv-Reg; RL
GPT-C2F (Chen et al., 2020a)	GPT2-*small*/*medium*	117/345M	MLE
Fine-tuned GPT2 (Chen et al., 2020b)	GPT2-*small*	117M	FT
PLOG (Liu et al., 2022)	BART-*large*	406M	PT/FT
	T5-*base*/*large*	220/770M	PT/FT
REASTAP (Zhao et al., 2022)	BART-*large*	406M	PT/FT
DEVTC (Perlitz et al., 2022)	GPT2-*small*/*medium*	117/345M	Supervised
LoFT (Zhao et al., 2023b)	BART-*large*	406M	Supervised
LOGEN (Deng et al., 2023)	GPT2-*small*	117M	Few-shot self-training
DistilTBR (Yang et al., 2024)	Flan-T5-CoT-*base*/*large*	250/780M	FT
	T5-CoT-*base*/*large*	220/770M	
RKT (Liu et al., 2024)	BART-*large*	406M	FT
RKF (Bai et al., 2025)	BART-*large*	406M	FT

Table 3.3: Pre-trained language models fine-tuned for LT2T. MLE = Maximum Likelihood Estimation, Adv-Reg = Adversarial Regularization, RL = Reinforcement Learning, PT = Pre-training, FT = Fine-tuning.

Method	Base Model	Param. Size	Training Method
MURMUR (Saha et al., 2023)	OPT	175B	In-context learning
GPT-3.5 (Zhao et al., 2023d)	GPT-3.5	175B	In-context learning
GPT-4 (Zhao et al., 2023d)	GPT-4	-	In-context learning
RKT (Liu et al., 2024)	LLAMA2	6.7B	FT
DistilTBR (Yang et al., 2024)	GPT-3.5-turbo	175B	CoT Distillation
RKF (Bai et al., 2025)	GPT-4o	-	Distillation

Table 3.4: LLMs used via prompting for LT2T.

PLMs, such as BART (Lewis et al., 2020), T5 (Qader et al., 2018), or GPT (Radford et al., 2018) on one or more LT2T datasets. Because these models have strong linguistic priors, only light task-specific tuning is required to outperform scratch baselines and traditional methods. Table 3.3 represents methods that utilize PLMs in their architecture.

Large Language Models (LLMs): A growing line of work leverages LLMs such as OPT (Zhang et al., 2022) and GPT-3/4 (Brown et al., 2020; Bubeck et al., 2023) without task-specific fine-tuning. Instead of updating weights, researchers steer models via in-context learning, including zero- or few-shot exemplars (Brown

et al., 2020), and Chain-of-Thought (CoT) prompting (Wei et al., 2022a) to boost faithfulness.

Beyond prompting, LLMs act as teachers, synthesizing CoT rationales to train smaller students (DistilTBR) or to transfer reasoning knowledge (RKT), and as automated annotators whose labels supervise downstream classifiers (RKF). Table 3.4 summarizes these LLM-based strategies.

3.5 Challenges in LT2T

This section discusses key challenges that have been addressed by existing research, particularly to improve LT2T.

Logical Fidelity: A generated sentence has *perfect fidelity* when every conclusion it makes is entailed by the information in the table. This means that each claim must logically and necessarily follow from the premises provided in the table. If a conclusion cannot be derived solely from the information presented, the text lacks fidelity. Common causes of low fidelity include: (1) missing or incorrect logical operations, (2) a mismatch between the sequence and logical order, and (3) over-reliance on superficial correlations rather than causal relationships grounded in the table.

Logical Controllability: The space of possible valid descriptions is exponentially large, as multiple logical inferences can be made from the same table. Current models struggle to determine which logical operation to apply, leading to irrelevant or uncontrolled outputs. Without guidance, models may select incorrect logic or reasoning paths. Therefore, it matters how models decide what content to include when multiple valid statements can be derived from a table.

Numerical Reasoning: Models commonly used in T2T, such as Pre-trained

Language Models (PLMs) or general-purpose Large Language Models (LLMs), are trained as text token predictors rather than explicit numerical reasoners. As a result, they often exhibit weaknesses in tasks that require one or more of these numeric competencies: (1) *Magnitude Understanding*: Magnitude indicates the size of a number and is used to compare or order values; (2) *Arithmetic/Functional Operations*: Performing exact or approximate operations such as addition, subtraction, ratios, and aggregates; (3) *Number to Word Mapping*: Choosing appropriate lexical descriptors.

Data Scarcity: The success of many LT2T methods depends on the availability of large amounts of annotated data. However, annotation is an expensive task. Therefore, developing methods to enable text generation with few-shot samples and reducing annotation costs are critical.

Diversity: Models often focus on the same table regions or apply the same logical operations, leading to repetitive outputs. To promote diversity, it is essential to generate multiple distinct yet factually accurate statements derived from a table.

User Preference: Different users may want different logical views, such as trends vs. outliers. Ignoring this leads to mismatched summaries. Conditioning generation on user intent improves relevance and reasoning selection.

Evaluation Methodology: Conventional automatic metrics such as BLEU (Papineni et al., 2002) reward surface-level token overlap, so they miss logical inconsistencies and hallucinations. They also lack explanations as to why outputs are (in)correct. Consequently, the field needs logic-aware, explainable metrics that can both assess and justify a model's reasoning accuracy.

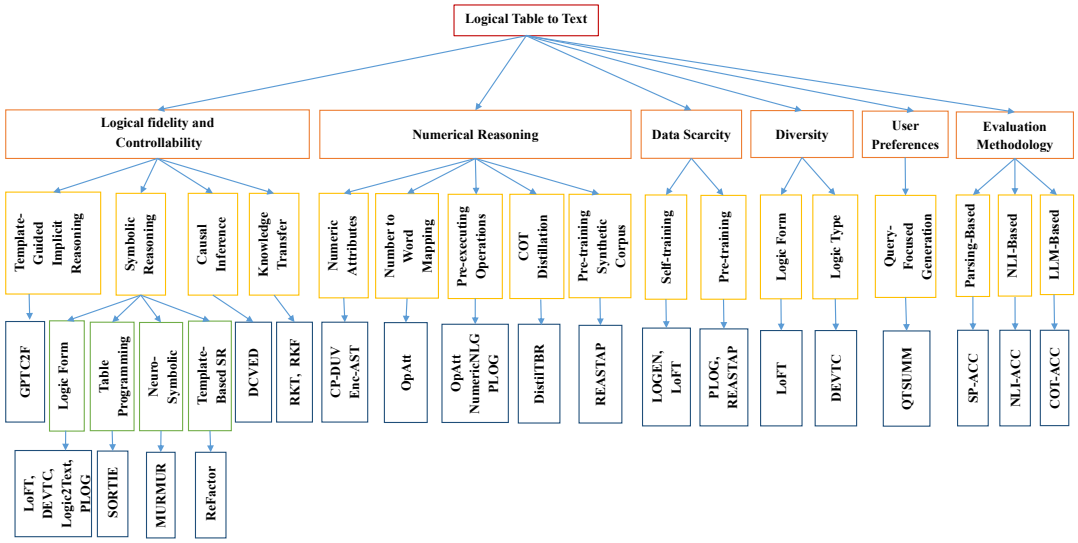


Figure 3.2: Taxonomy of logical table-to-text methods

3.6 Methods for Logical Table-to-Text

This section groups methods by the challenges they address; each method might appear in multiple categories. Figure 3.2 visualizes this: orange boxes mark key issues from Section 3.5, yellow and green boxes show main method families, and blue boxes list representative approaches. The diagram also outlines the order of the subsections that follow.

3.6.1 Logical Fidelity and Controllability

This section reviews papers that propose approaches to enhance logical fidelity and faithfulness in LT2T. Because many of these methods simultaneously enhance controllability, we discuss approaches for both issues here.

Template-Guided Implicit Reasoning

Auto-regressive generators emit tokens strictly left-to-right; however, logical reasoning can require performing step C before step B. This misalignment between language order and logical order can introduce reasoning errors (Chen et al., 2020a). To address this issue, Chen et al. (2020a) propose a two-stage coarse-to-fine decoder known as GPT-C2F. First, the model generates a template by replacing entities and numbers with placeholders to plan the logical structure. After fixing the global structure, it generates the final sentence by conditioning on both the table and its template to select the correct entity or number for each placeholder. Because the second pass treats the entire template as fixed context, each placeholder attends to its left and right neighbors and the connective words. This allows the model to infer the intended operation before selecting a value, thereby reducing order-mismatch errors.

Symbolic Reasoning-Based Methods

Symbolic reasoning (SR) introduces an explicit intermediate representation, such as logic forms (LFs), programs, or modular operators that specify exact operations to perform before text generation. By dividing a reasoning task into a sequence of clear, executable subtasks, SR approaches can validate each step in isolation and discard logically unsound paths before generating text. Because the final output is grounded in a verified chain of operations, these methods substantially improve logical fidelity and overall faithfulness.

Logic-Form–Based Methods: This subsection reviews methods that utilize logic-forms during different phases of their approach, such as pre-training, training, or inference to address fidelity.

Logic2Text (Chen et al., 2020b) and LoFT (Zhao et al., 2023b) both condition text generation on a table and its logic form. In both methods, the LF guides which table region and operation to reference, reducing ambiguity, and improving control over content selection. Logic2Text uses gold, hand-annotated Python-style LFs whose execution has already been validated on the table; no extra checking is required. However, LoFT builds LFs automatically via a Structure-Aware Semantic Parsing (SASP) approach (Ou and Liu, 2022) at training time and a synthesis pipeline (Liu et al., 2022) at inference. Because automatically generated LFs can be noisy, LoFT also treats them as fact verifiers: each candidate LF must execute to True, and a Natural Language Inference (NLI)-based method filters out any sentence that the table does not entail.

PLOG (Liu et al., 2022) utilizes LFs as pretraining signals to teach logical reasoning. The idea is that by first training LFs, the model develops a deeper understanding of logical structures, which results in fewer logical errors in final text generation. In pretraining, a large synthetic dataset of (table, LF) pairs is used. The model is then fine-tuned to generate statements from tables. This approach improves controllability by explicitly highlighting which table cells are involved in each LF.

Table-Compatible Programming Language: SORTIE (Zhao et al., 2023a) improves logical reasoning by decoupling language generation from reasoning. It builds on Chen et al. (2020a)'s two-stage approach of generating a template, but fills each template placeholder via symbolic program execution instead of neural prediction. For every placeholder, a Gated Recurrent Unit (GRU) (Chung et al., 2014) scheduler picks an order based on logical dependencies, and another GRU generates a short, table-compatible program composed of operators and operands whose execution on the table returns the exact value and prevents hallucination.

To handle missing annotations, SORTIE uses heuristic pseudo-labels with self-adaptive training. This clear division between *what to say* via template and *how to reason* by programs enhances logical fidelity and faithfulness.

Neuro-Symbolic Modular Reasoning: MURMUR (Saha et al., 2023) is a neuro-symbolic, modular framework designed to improve logical reasoning by explicitly separating symbolic logical reasoning from linguistic generation. Its core idea is to dynamically construct executable reasoning paths composed of symbolic and neural modules, governed by grammar and guided by a saliency-based value function during a best-first search. By explicitly performing logical operations using symbolic modules and restricting permissible compositions via grammar rules, MURMUR ensures each reasoning step is valid and verifiable. This approach directly addresses pitfalls such as hallucination, order mismatches, and semantic inconsistencies. Once a valid reasoning path is constructed, a language model converts it into a natural sentence. This method improves both the accuracy and faithfulness of generated summaries by making the reasoning process explicit and reliable.

Template-Based Symbolic Reasoning: ReFactor (Zhao et al., 2023c) explicitly retrieves and generates fact-guided reasoning signals. A set of predefined templates is designed to target reasoning skills such as numerical comparisons, aggregations, and conjunctions. These templates are instantiated and executed over tables using a fact generator to produce multiple facts. Relevant facts are ranked based on user input and included as signals to the model. This improves factual accuracy by providing explicit symbolic reasoning during generation.

Causal Inference

DCVED (Cheng et al., 2021) addresses logical inconsistency caused by *hidden confounders*, unobserved factors that create spurious correlations between the input table and the output text (Keith et al., 2020). To resolve this, DCVED applies causal intervention using do-calculus (Pearl, 2010), shifting the learning objective from $p(y \mid x)$ to $p(y \mid \mathrm{do}(x))$, thereby reducing the influence of confounders. To implement this, DCVED frames the generation process using a causal graph with two key variables: (1) Mediator z_m: information extracted from the table, (2) Confounder z_c: a variational latent representing misleading patterns such as frequent but irrelevant table entities. These latent spaces are supervised to be meaningful: z_m is guided by entities mentioned in the target sentence, while z_c is trained to predict unused but high-frequency distractors. During inference, DCVED samples multiple sentences by varying z_c and uses a trained model to select the most factually consistent output. By combining causal reasoning with variational modeling, spurious correlations are reduced, and logical fidelity is improved.

Knowledge Transfer

Liu et al. (2024) employs Reasoning Knowledge Transfer (RKT) to improve logical fidelity in LT2T. They fine-tune a LLaMA-2-6B as a transfer model on Logic2Text to produce natural-language logical rules from tables, synthesize such rules for LogicNLG, and then train a two-stage BART system: a reasoning module that predicts rules from tables and a summary module that generates text conditioned on those rules—making outputs table-entailed rather than correlation-driven. Noting that some transferred rules are noisy, Bai et al. (2025) introduces

the Reasoning Knowledge Filter (RKF) that employs a clean-up stage. RKF uses GPT-4o to annotate a subset of the training data for reasoning correctness. This subset is used to train a BART-large classifier that filters low-quality reasoning traces from the rest of the dataset. This filtering process improves the logical fidelity by ensuring the generator only sees high-quality, table-consistent reasoning paths.

3.6.2 Numerical Reasoning

This section reviews methods that address numerical reasoning issues discussed in Section 3.5.

Understanding Numeric Attributes

CP-DUV (Gong et al., 2020) targets neural models' poor grasp of numerical magnitude caused by treating numbers as word tokens through two key upgrades: (1) They inject magnitude-aware embeddings. A transformer is pre-trained to rank every pair of numbers within a column; each number token is replaced at runtime by an embedding that knows both its size and realistic context. (2) They add a content-aware verification reward—a policy-gradient signal that rewards summaries containing correct entities and statistics in logical order while penalizing omissions and redundancies. These additions give the model a true *sense of numbers*, reducing magnitude errors and ordering output.

Enc-AST (Li et al., 2021) enhances numerical reasoning by addressing magnitude, relative importance, and inter-entity relationships. A hierarchical encoder incorporates two auxiliary tasks: (1) *Number ranking* captures column-wise magnitude; (2) *Importance ranking* models row-wise importance of numerical values.

These tasks are learned through self-attention and fused via a gating mechanism. Additionally, a graph-based reasoning module models relationships between entities and enables reasoning over inter-entity relationships. These components enhance the model's ability to generate factually accurate and numerically grounded summaries.

Number to Word Mapping

OpAtt (Nie et al., 2018) addresses the issue of mapping numbers to words using a quantization layer that groups scalar values into a small number of learnable bins. A linear layer maps numbers into logits, then a softmax layer converts the logits into a probability distribution over bins, each with its own embedding. The final embedding is a weighted sum of all embeddings. This method enables the model to generalize across similar values and produce magnitude-aware words during text generation.

Pre-executing Operations

These methods pre-execute numerical operations and feed the results to the model to enhance its numerical reasoning. This reduces the burden of calculation and turns numeric operations that neural LMs struggle with into plain sequence tokens or features that models can copy or attend to.

OpAtt (Nie et al., 2018) pre-executes operations such as *minus* for score gaps and *argmax* for identifying the top scorer. The operations and results are encoded beside records and fed into the decoder. The decoder uses dual attention over operations and records, with a gating mechanism to decide when to prioritize operations or records. Additionally, the copy mechanism ensures outputs can originate from table cells, improving faithfulness.

Suadaa et al. (2021) pre-computes *max*, *min*, and *diff* for the target rows, stores them in a dedicated operation table (T_{OP}), and trains a copy-augmented model that uses placeholders. At inference, a ranking-and-memory procedure replaces each placeholder with a value drawn from T_{OP} or the original data table, ensuring every numeric value in the output is faithfully copied rather than hallucinated.

PLOG (Liu et al., 2022) computes column-wise *sum*, *avg*, and *per-cell rank* values and inserts them into the flattened table structure, allowing the model to access them naturally during generation.

These methods demonstrate that pre-execution, alongside attention, gating, or copying, substantially improve models' numerical reasoning.

Chain-of-Thought Distillation

DistilTBR (Yang et al., 2024) transfers numerical-reasoning skills from an LLM to smaller PLMs by first having the teacher model produce step-by-step reasoning traces for each table. These traces, appended to the table, serve as supervised signals when fine-tuning the student model, guiding it to look at the right rows/-columns and apply the correct arithmetic operations. This distillation yields a compact generator that mimics the teacher's logical reasoning without altering the underlying architecture.

Pre-training with Synthetic Corpus

REASTAP (Zhao et al., 2022) injects diverse reasoning skills into a seq2seq model via synthetic QA pairs during pre-training, without relying on table-specific architectures. Conditioned on a serialized table and question, the model learns to align columns, filter rows, and execute numerical operations purely through its

parameters.

3.6.3 Data Scarcity

Several recent methods aim to reduce the reliance on large amounts of annotated data by generating supervision from unlabeled or synthetic sources.

One approach is to create synthetic corpora offline, allowing models to pre-train before being fine-tuned on limited gold data. For example, PLOG uses handcrafted logic-form templates filled with table values. Instances are kept only if the generated logic-form evaluates to True, producing a large, automatically verified dataset. Similarly, REASTAP (Zhao et al., 2022) creates synthetic question-answer pairs using a template-based example generator (Yoran et al., 2022) for various reasoning skills over tables. This synthetic generation allows the model to learn effectively without relying heavily on human-annotated data.

A second line of work focuses on self-training via pseudo-labeling, where models expand a small seed set using unlabeled data. LOGEN (Deng et al., 2023) begins with a tiny annotated set and trains two GPT-2 models: one to map tables and logic forms to text; one in reverse. The reverse model generates logic forms for new (table, text) pairs, and mutual checks retain consistent examples that are used to retrain the models in a bootstrap loop.

Finally, on-the-fly weak supervision eliminates the need for any gold logic forms. LoFT synthesizes multiple logic-form candidates per table at training and inference time. An NLI-based verifier filters out any that aren't supported by the table, allowing the model to learn from verified candidates without manual annotation.

3.6.4 Diversity

LoFT and DEVTC (Perlitz et al., 2022) both improve diversity by enumerating alternate reasoning structures. At inference, LoFT creates multiple candidate logic forms per table from 45 templates across eight operation categories, while DEVTC samples from various logic types. Because each logic form or type focuses on a different operation or table region, the models generate a broad set of distinct, valid statements.

3.6.5 User Preference

QTSUMM (Zhao et al., 2023c) extends LT2T by tailoring generated summaries to user preferences. To achieve this, they fine-tune models on their *QTSUMM* dataset of query–summary pairs. During training, the model is conditioned on user queries alongside the table, guiding generation and producing summaries that directly address user-specific information needs. Additionally, ReFactor (Section 3.6.1) generates query-relevant facts, which are concatenated to the input of the models or used as prompts in LLM during fine-tuning and inference. These facts provide explicit reasoning evidence, further guiding the model to address user queries in the summary with high analytical fidelity.

3.6.6 Evaluation Methodology

This section reviews automatic metrics for evaluating logical fidelity and explainability.

Parsing-Based Evaluation: This metric evaluates logical fidelity by converting each generated sentence into an executable logic form and running it against the source table. The generated text is converted to the candidate logic forms

using a weakly supervised semantic parser (Liang et al., 2009) via breadth-first search. The candidates are ranked based on consistency with the original sentence. The top-ranked logic form is selected and executed on the table. The generated statement is considered logically faithful if the result is `True`. *Semantic Parsing Accuracy* (SP-Acc) (Chen et al., 2020a) is the proportion of sentences whose top logical form evaluates to `True`. Its reliability depends on the parser quality; misparsing or language ambiguity can lead to false outcomes.

Natural Language Inference (NLI)-Based Evaluation: This metric verifies whether a given statement is true, false, or neutral based on the table. The key point is to utilize an NLI model trained to predict the logical relationship to measure the entailment score between the table and the generated text. The ratio of entailed is computed and used to approximate the model's fidelity. NLI-Acc (Chen et al., 2020a), TAPEX-Acc, and TAPAS-Acc (Liu et al., 2022) are examples of this evaluation metric.

Reference-Free, LLM-based Evaluation: CoT-Acc (Zhao et al., 2023d) uses a 2-shot chain-of-thought prompt with GPT-3.5/4 to label each generated sentence as entailed or refuted with respect to the table, then reports accuracy. It yields the best human correlation among automated faithfulness metrics on LogicNLG.

3.7 Comparing Methods

We compare representative methods across three key dimensions: performance, efficiency, and interpretability/controllability. Since our focus is on logical reasoning, we evaluate performance primarily using logical fidelity metrics in Table 3.5.

Pre-trained vs Traditional Backbones: Overall, pre-trained backbone mod-

Method	Backbone	Family	SP-Acc ↑	NLI-Acc ↑
Field-Infusing (Chen et al., 2020a)	Transformer	-	38.9	57.3
GPT-TabGen (med) (Chen et al., 2020a)	GPT-2	-	45.5	73.3
GPT2-C2F (med) (Chen et al., 2020a)	GPT-2	Template-Guided	45.3	76.4
DCVED (Cheng et al., 2021)	GPT-2	Causal Inference	43.9	76.9
DEVTC (Perlitz et al., 2022)	GPT-2	Logic Form	45.6	77.0
PLOG (Liu et al., 2022)	BART-Large	Logic Form	50.5	88.9
LoFT (Zhao et al., 2023b)	BART-Large	Logic Form	57.7	86.9
REASTAP (Zhao et al., 2022)	BART-Large	Pretraining Synthetic	54.8	89.2
SORTIE (Zhao et al., 2023a)	BART-Large	Table Programming	57.8	89.3
RKT (Liu et al., 2024)	BART-Large	Knowledge Transfer	59.6	88.1
RKF (Bai et al., 2025)	BART-Large	Knowledge Transfer	61.0	88.8

Table 3.5: Comparison of LT2T Models on Logical Fidelity Metrics. Shows the best results reported in the original papers.

els consistently outperform traditional encoder–decoder architectures. For example, even TabGen, which is only fine-tuned on LogicNLG, surpasses field-infused Transformer models in logical fidelity metrics.

Template-Guided Implicit Reasoning: GPT-C2F, which utilizes coarse-to-fine generation, yields higher NLI-Acc and Adv-Acc than GPT-TabGen at both small and medium scales.

Logic-Form/Logic-Type Control: Human evaluations (Chen et al., 2020b) reveal a dramatic gain in factual correctness when logical forms are used, jumping from 20% without LFs to over 82% with LFs. DEVTC improves on prior GPT-2 baselines by conditioning generation on predicted logic types. While it gains slightly in logical metrics, its main strength lies in diversity: by switching logic types during generation, it achieves a factuality–diversity trade-off that surpasses the GPT-TabGen sampling frontier without requiring stochastic decoding, using greedy decoding alone. LoFT builds on this by conditioning on full executable logic forms and applying a verifier to filter outputs. This enables LoFT to achieve the best overall balance between faithfulness and diversity in this family. The

trade-off, however, is increased implementation complexity: DEVTC requires a logic-type classifier, while LoFT involves a full pipeline with logic-form parsing, synthesis, and verification. These methods offer improved controllability and more transparent reasoning processes.

Implicit Skill Injection (Pretraining): PLOG and REASTAP inject logical reasoning through pretraining; PLOG leverages logical forms (LFs), while REASTAP uses 4 million synthetic question–answer pairs. Both significantly improve logical fidelity metrics. These approaches keep inference simple at the expense of pretraining cost; for instance, REASTAP's pretraining required 34 hours on an 8 NVIDIA A5000 24GB cluster. They improve faithfulness through pretraining signals, but the final generation remains purely neural, so the internal reasoning steps aren't explicitly exposed.

Table Programming: SORTIE reports the strongest NLI-Acc among compared methods. It outperforms PLOG and REASTAP without any large-scale reasoning pretraining—training for roughly 5 hours on 8×3090 Ti GPUs—while offering high interpretability via explicit, executable reasoning traces. The trade-off is a higher implementation burden.

Knowledge Transfer: RKT and RKF report the strongest logical-fidelity metrics among peers. Both are highly interpretable and controllable via explicit rules.

Not all models are evaluated on these metrics; we discuss other models that use other logical metrics in the following.

3.7.1 Other evaluation metrics

We summarize the best results of models evaluated with TAPAS-Acc and TAPEX-Acc in Table 3.6, using the values reported in their original papers.

Method	Backbone	Family	TAPAS-Acc↑	TAPEX-Acc↑
PLOG (Liu et al., 2022)	T5-Large	Logic Form	76.0	75.9
LoFT (Zhao et al., 2023b)	BART-Large	Logic Form	-	61.8
LLM-T2T (Zhao et al., 2023d)	GPT-4-Zero-shot	LLM	91.8	91.0
	GPT-4-1-shot-direct		87.6	88.0
	GPT-4-1-shot-COT		89.4	90.8
	GPT-4-2-shot-direct		92.0	89.6
	GPT-4-2-shot-COT		88.8	90.4
ReFactor (Zhao et al., 2023c)	GPT4-zero-shot	Template-Based SR	92.3	-
	GPT3.5-1-shot		94.3	-
	GPT4-2-shot		93.3	-
DistilBTR (Yang et al., 2024)	FLan-T5-Base-COT	COT Distillation	78.72	82.75
	T5-Large-COT		80.62	81.97

Table 3.6: Comparison of LT2T Models on other logical fidelity Metrics.

COT Distillation/LLMs: LLM-T2T (Zhao et al., 2023d) reports that GPT-family models outperform fine-tuned systems such as logic-form-based methods, including LoFT and PLOG. It also finds that adding more shots or Chain-of-Thought (CoT) prompting yields non-monotonic gains—more exemplars or CoT do not necessarily help; GPT-4-zero-shot performs better than GPT-4-1/2-COT. Zhao et al. (2023c) attain state-of-the-art TAPAS-Acc by augmenting LLMs with ReFactor. Finally, DistilTBR demonstrates that distilling CoT-style supervision into smaller models like Flan-T5 and T5 can improve logical fidelity without relying on very large LLMs. MURMUR shows via human evaluation 26% more logically consistent summaries on LogicNLG vs direct prompting.

We further provide a comparative rating of all surveyed methods along five axes including diversity, interpretability, controllability, implementation burden, and cost in Table 3.7. The accompanying text details the rubric and justifies each assignment. In the following, we explain how we rank each column in Table 3.7.

Diversity

Low: The model repeatedly generates similar outputs that rely on the

Method	Diversity	Controllability	Interpretability	Implementation Burden
GPT-C2F	Med	Med	Med	Med
Fine-tuned GPT-2	Med	High	High	Med
DCVED	Low	Low	Low	Med
PLOG	Med	Med	Low	Med
REASTAP	Med	Med	Low	High
DEVTC	High	Med	Med	Med
LoFT	High	High	High	High
LOGEN	Med	Med	Med-High	Med
MURMUR	High	High	High	High
ReFactor	Med	Med	Med	Med
LLM-T2T	Med	Med	Med	Low
DistilBTR	Low-Med	Low	Med	Med
RKT	Med	High	High	Med-High
RKF	Med	High	Med-High	Med

Table 3.7: Comparing methods based on various criteria.

same logical operation or remains confined to a fixed table region.

Medium: The model produces varied statements, but often focuses on similar regions or operations unless explicitly guided.

High: The model includes explicit mechanisms to diversify content selection, such as logic-type control, logic-form conditioning, or prompt variation, enabling it to describe different logic types and table regions.

Controllability

Low: There is little or no means to steer logical operations; the model explores a large, unconstrained search space.

Medium: Logic-type tokens, templates, or other signals allow partial steering of logic types but not specific operations or arguments.

High: The method allows explicit control via intermediate structures, including logic forms, programs, or verified candidates that deterministically guide generation.

Interpretability

Low: No explicit intermediate reasoning artifacts are available; the model operates as a black box.

Medium: Lightweight signals such as logic-type tags, CoT summaries, or templates provide some interpretive cues but not complete reasoning traces.

High: The model produces executable or inspectable intermediates such as logic forms or symbolic programs that make the reasoning process transparent.

Implementation Burden

Low: Involves only prompt design or minimal fine-tuning.

Medium: Requires one auxiliary component (e.g., classifier, teacher model, or template mechanism) within a standard training pipeline.

High: Entails multi-stage pipelines (e.g., parser/synthesizer, generator, and verifier) or large-scale pretraining/data synthesis, often with custom executors.

3.8 Datasets and Evaluation Metrics

Method	Dataset	Surface Metric	Logic Metric	Human Eval.
OpAtt	RotoWire	BLEU	RG	–
CP-DUV	RotoWire; MLB	BLEU	RG / CS / CO	–
Field-Gating	LogicNLG	PPL; BLEU-1,2,3	SP-Acc; NLI-Acc; Adv-Acc	Yes
Field-Infusing	LogicNLG	PPL; BLEU-1,2,3	SP-Acc; NLI-Acc; Adv-Acc	Yes
DCVED	LogicNLG; Logic2Text	BLEU	SP-Acc; NLI-Acc	–
BERT-TabGen	LogicNLG	PPL; BLEU-1,2,3	SP-Acc; NLI-Acc; Adv-Acc	No
GPT-TabGen	LogicNLG	PPL; BLEU-1,2,3	SP-Acc; NLI-Acc; Adv-Acc	Yes
GPT-C2F	LogicNLG	BLEU-1,2,3	SP-Acc; NLI-Acc; Adv-Acc	Yes
Fine-tuned GPT2	Logic2Text	BLEU-4; ROUGE-1,2,L	No	Yes
PLOG (BART/T5)	LogicNLG; CONTLOG	BLEU-1,2,3	SP-Acc; NLI-Acc; TAPEX/TAPAS-Acc	Yes
REASTAP	LogicNLG	BLEU-1,2,3	SP-Acc; NLI-Acc	No
DEVTC	LogicNLG	BLEU-1,2,3	SP-Acc; NLI-Acc	Yes
LoFT	LogicNLG	BLEU-1,2,3	SP-Acc; NLI-Acc	Yes
LOGEN	Logic2Text	BLEU; ROUGE-L	No	Yes
MURMUR	LogicNLG	BLEU; METEOR	No	Yes
DistilTBR	SciGen	METEOR; BERTScore; BLEURT	TAPAS/TAPEX-Acc	No

Table 3.8: Datasets and evaluation metrics used in surveyed papers.

Table 3.8 shows the datasets and evaluation methodology used. Note that early studies evaluated their methods on datasets that are not specifically designed for LT2T, including RotoWire and MLB.

3.9 Discussion

This survey found two dominant LT2T paradigms: (1) **Explicit trace construction:** logic-forms, programs, or neuro-symbolic plans that decouple reasoning from surface realization and give strong fidelity guarantees. (2) **Implicit skill injection:** pretraining or distillation schemes that embed numerical or logical competence inside neural parameters with minimal task-specific engineering.

Explicit methods appear better when one needs: (a) auditability/controllability; (b) verifiable, step-wise reasoning; or (c) operator-level control. However,

we suggest *implicit methods* when (a) simple, fast inference at scale is required; (b) there is lack of annotations or tooling for logic forms; or (c) one needs broad skill coverage from synthetic data.

3.10 Future Directions

We suggest four primary avenues for advancing LT2T research, considering current limitations and existing challenges.

Improving Reasoning Capabilities: Existing approaches are limited to a few operations, such as *sum*, *min*, or *max* (Appendix A.1), and struggle with more advanced analytical needs, such as trend detection or multi-hop comparisons. Research should address this limitation through three main thrusts: (1) Support more complex logical skills for advanced analytics, including change rates, graph traversal, and temporal patterns. (2) Develop flexible logical supervision frameworks that help models acquire these skills without explicit structured logical forms. Lightweight, weakly supervised, or self-training methods may allow models to infer logical structures from training data without requiring gold-standard logic annotations. (3) Include dynamic logical inference, where the model constructs and executes operator chains during inference to ensure faithful intermediate results before verbalization. Approaches may range from symbolic search with learned value functions to prompting LLMs to produce executable code. Together, these directions will enable broader coverage of analytical reasoning skills, stronger factual accuracy, and adaptability to evolving data schemas.

Evaluation Metrics: We need metrics that are both logic-aware and explainable. While metrics like COT-Acc offer step-by-step reasoning to judge entailment, they are costly, non-deterministic, and limited to proprietary APIs. We

need open, lightweight, logic-aware metrics that break outputs into table-grounded claims and mark each as *entailed*, *contradicted*, *missing*, or *hallucinated*. Moreover, existing metrics for logical reasoning report aggregate scores that obscure which operations succeed and where models fail. These metrics cannot determine whether the generated sentence actually matches the logic type requested by the user.

Scalability: Future LT2T systems must adapt to evolving computational resources and table schemas. New schemas often demand novel logical skills, such as rate-of-change analysis or multi-row trend detection, which current models struggle to acquire without retraining. Exploring continual or incremental learning frameworks that allow models to acquire new reasoning skills while retaining prior knowledge appears promising. Moreover, the high training and inference costs of LLMs, together with their fixed context-window limits, make them impractical for many real-world deployments. Research should thus prioritize lightweight, modular, or distilled models that retain strong logical-reasoning capabilities while reducing computational and memory requirements.

Applicability: Current models are designed for flat tables, whereas real-world data often involves heterogeneous formats like event sequences, hierarchical tables with multi-level headers, or nested structures. Future LT2T systems should be able to process complex structured data and reason over temporal, hierarchical, or relational formats, as seen in domains like medicine or finance. There is also a need for multi- and cross-lingual LT2T generation, where models maintain logical reasoning consistency across languages. Logical operations such as arithmetic, comparison, causal inference, and multi-hop deduction must transfer cleanly across linguistic boundaries. Expanding LT2T to low-resource languages, global reporting, and multilingual scientific domains will promote greater robust-

ness, generalization, and cross-domain utility.

3.11 Conclusions

Logical Table-to-Text (LT2T) has become a microcosm of reasoning-aware generation for structured inputs, explicit operations, and verifiable outputs. Explicit methods boost fidelity and interpretability, while implicit ones scale better; hybrid models that pair symbolic checks with distilled reasoning show promise in balancing both.

Though focused on LT2T, these insights extend to broader text generation: reasoning traces, logic-aware supervision, and fidelity-based evaluation are key for summarization, retrieval-augmented, and multi-modal tasks. Future progress lies in richer operator coverage, explainable metrics, and lightweight continual-learning frameworks.

Chapter 4

Operation-Level Logical Fidelity in Table-to-Text Generation

Logical Table-to-Text (LT2T) generation aims to produce natural-language sentences that are logically faithful to structured tabular data. While recent Large Language Models (LLMs) show high performance on aggregate fidelity metrics, these scores provide only a coarse view of performance, obscuring specific failures in logic-type reasoning and models' meta-logical awareness. We propose an operation-aware diagnostic framework that evaluates four core competencies: (1) Logical Form (LF) execution accuracy, (2) fidelity of LF-conditioned generation, (3) logic-type identification, and (4) LF-free same-type generation.

We apply this framework to a suite of LLMs and perform fine-grained analysis across logic types such as Aggregation, Ordinal, and Superlative reasoning. Our results show that LT2T fidelity assessment can be unstable; the choice of verifier and logic type can substantially alter conclusions and model rankings. Crucially, we identify a meta-logical gap: models often generate faithful statements while failing to identify the underlying operation.

4.1 Introduction

Logical Table-to-Text generation aims to produce natural-language descriptions that are both fluent and logically faithful to structured tabular data. Unlike surface-level data-to-text generation, LT2T requires explicit reasoning over table entries, including comparison, aggregation, counting, and arithmetic operations. Ensuring logical fidelity is critical for real-world applications such as scientific reporting, financial summarization, and decision support, where even minor reasoning errors can lead to misleading or incorrect conclusions.

Recent advances in LLMs have led to substantial improvements across a wide range of natural language processing tasks, including reasoning-intensive generation. LLMs demonstrate strong performance in LT2T settings through in-context learning and chain-of-thought prompting (Zhao et al., 2023d), often achieving high scores on existing logical fidelity metrics without fine-tuning. However, these gains are typically reported at an aggregate level, providing limited insight into the models' underlying reasoning behavior or their performance on specific logical operations.

We argue that relying solely on an aggregate score obscures critical failure modes. In practical applications such as data journalism or dashboard narration, users depend on the reliability of specific operations, such as identifying the largest value or calculating counts. A model may achieve a high average score yet remain fundamentally unreliable for specific, decision-critical logic types such as Ordinals or Aggregation. Furthermore, existing evaluations fail to reveal the source of these errors: is it a failure in grounding, operator selection, or linguistic realization? Without this granularity, we lack the diagnostic clarity needed to build more robust systems or implement targeted fixes.

In this work, we move beyond aggregate evaluation to present an operation-aware diagnostic analysis of LLM-based LT2T systems. We systematically investigate model behavior across diverse logical operations through four research questions:

(RQ1) Execution How accurately can models execute logical forms (LF)s against tables?

(RQ2) Fidelity How faithfully do models realize a given LF into a natural language sentence?

(RQ3) Identification Can models recognize the logic type implied by the generated statement?

(RQ4) Generalization Can models generate a new statement of the same logic type without access to the logical form?

We translate these questions into a suite of targeted tasks and evaluate models per logic type to uncover operation-specific strengths, failure modes, and scaling trends that are invisible under aggregate fidelity scores.

Our operation-aware analysis shows that high aggregate LT2T scores can mask important weaknesses. We highlight two recurring phenomena. First, we observe a **meta-logical gap**: models may generate claims that appear faithful to the table while still failing to correctly identify the underlying reasoning operation. This suggests that accurate outputs do not necessarily reflect robust operator-level understanding. Second, we find **metric instability**: conclusions about performance, including model rankings, can vary substantially depending on the verifier used, indicating that current metrics capture different and sometimes conflicting aspects of logical fidelity.

4.2 Related Work

Logical Table-to-Text Generation. Logical Table-to-Text (LT2T) generation extends the table-to-text task by requiring models to realize explicit logical reasoning, such as comparison or counting, as faithful natural language statements. Following the taxonomy established in Chapter 3, existing approaches address this requirement either by explicitly conditioning on symbolic structure, such as logical forms (Chen et al., 2020b) or programs executed over the table (Zhao et al., 2023a), or by implicitly inducing reasoning behaviors through pretraining (Liu et al., 2022) or distillation (Yang et al., 2024). More recently, studies on LT2T with large language models (LLMs) report strong overall performance (Zhao et al., 2023d). However, aggregate metrics can mask sharp operation-specific failures; systematic, operation-aware analysis by logic type remains underexplored.

Evaluating Logical Fidelity in Table Reasoning. A significant body of evaluation work in table reasoning and LT2T treats fidelity as table entailment, utilizing NLI-style verifiers. In this paradigm, a model predicts whether a generated statement is entailed by the table, using the entailment rate as a proxy for logical fidelity. Examples include NLI-Acc (Chen et al., 2020a) and table-aware variants like TAPAS-Acc and TAPEX-Acc (Liu et al., 2022). While these approaches are robust to linguistic paraphrasing, they overlook fine-grained operator or argument errors and remain sensitive to how the table is linearized.

Complementarily, Semantic Parsing (SP)-based evaluation (Chen et al., 2020a) attempts to map the generated sentence back to a formal program or logical form. By executing this LF against the table, this method enforces a stricter notion of

program recoverability. However, this approach is inherently brittle; it frequently penalizes valid linguistic variations and depends heavily on the parser's ability to resolve ambiguity.

Considering that these verifier families capture distinct dimensions of correctness, our study provides operation-level diagnostics to pinpoint where models, and the verifiers themselves, succeed or fail.

4.3 Task Setup and Problem Formulation

In this section, we first introduce the problem and the logic types, and then define the set of tasks the models are prompted to complete in Section 4.3.3.

4.3.1 Logical Table-to-Text and Logical Forms

We study *Logical Table-to-Text* (LT2T) generation that is defined in 3.2.1.

A *logical form*, denoted p, represents the multi-step reasoning required to derive an answer from the table. Executing p on T yields a final value v such as a scalar or boolean.

4.3.2 Logic Type Taxonomy

To support operation-level analysis, we assign each LF/sentence to a *logic type* $l \in L$, where L includes: *Count*, *Aggregation*, *Superlative*, *Ordinal*, *Comparative*, *Majority*, and *Only*.

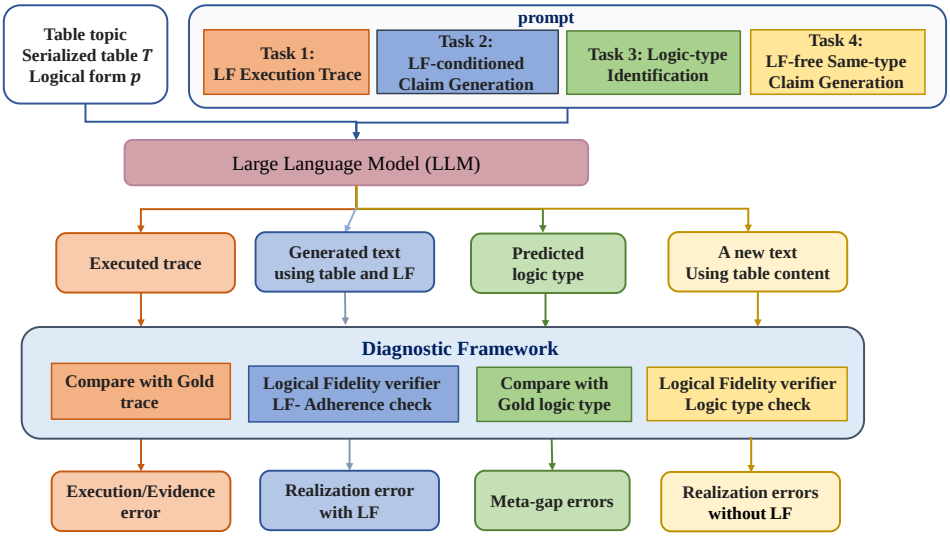


Figure 4.1: Operation-aware diagnostic framework

4.3.3 Task Definitions

To answer our research questions, we propose the diagnostic framework illustrated in Figure 4.1. It consists of four tasks: (1) LF execution and evidence grounding, (2) LF-conditioned generation fidelity, (3) logic-type recognition, and (4) LF-free type-controlled generation.

Task 1 (Execution Trace): This task examines a model's ability to execute an LF and ground each intermediate step in table evidence. Given a table and a logical form, the model outputs a nested execution tree τ aligned with the LF structure. For each operator node, the trace records the function name, its arguments, including nested subcalls, intermediate results, the input/output row sets, and the evidence cells used. This task captures **where** the model sources its evidence and enables step-level error attribution, such as incorrect

filter conditions versus aggregating over the wrong column.

Task 2 (Entailed Claim Generation): This task measures a model's ability to generate an entailed logical sentence when an LF is available. Given a table and a logical form, the model generates a single sentence y that expresses the final result of executing p on T. This task evaluates whether a model can faithfully **realize** a given LF in text by expressing the computed value(s) while respecting the intended operator semantics and arguments (columns, rows, constants), without introducing unsupported facts.

Task 3 (Logic-Type Identification): Given the table and the generated statement in Task 2, the model predicts a logic type label $l \in L$. This evaluates operation recognition, such as *Ordinal* vs. *Superlative*, not captured by standard fidelity scores.

Task 4 (LF-Free Claim Generation): Given the table, the model must generate a new entailed sentence y' that is grounded in a different part of the table, without access to a logical form. This task tests whether the model can generate an entailed sentence without access to an LF.

4.3.4 What These Tasks Enable

Together, the four tasks separate failure sources that are blurred by aggregate metrics: (1) *execution/evidence errors* (Task 1), (2) *LF-following realization errors* (Task 2), (3) *logic type recognition errors* (Task 3), and (4) *realization without LF errors* (Task 4). This formulation supports an operation-level diagnostic framework that can attribute errors to specific operations, arguments, and evi-

dence rather than reporting only end-to-end correctness.

4.4 Experiments

We first describe the test set used in our experiments in Section 4.4.1, followed by the evaluation metrics in Section 4.4.2. In Section 4.4.3, we report and analyze the automatic evaluation results of our research questions.

4.4.1 Test Set

We build a 700-instance test set by sampling from the Logic2Text development and test splits, comprising 100 instances for each logic type. Model outputs are grouped by their gold logic type, and results are reported separately for every type.

4.4.2 Automated Logical Fidelity Metrics

As discussed in Section 4.2, we evaluate sentence-level logical fidelity using two common families of automatic metrics: (1) NLI-style entailment verification, including NLI-Acc, TAPAS-Acc, and TAPEX-Acc. (2) Parsing-based execution: Sp-Acc.

4.4.3 Evaluation Results

We present results organized by our research questions, corresponding to Tasks 1–4. Unless stated otherwise, we report the scores of the above metrics (1) per logic type; (2) the macro-average across logic types for each model to summarize

Model	Aggregation	Count	Ordinal	Comparative	Only	Superlative	Majority	Average
GPT5.1	71.0	84.0	67.0	84.0	83.0	85.0	91.0	80.7
DeepSeek-V3.2	92.0	87.0	88.0	91.0	94.0	92.0	92.0	90.9
Qwen3-235B-A22B	88.0	87.0	**95.0**	**98.0**	92.0	92.0	94.0	92.3
Llama3.1-405B	82.0	**92.0**	91.0	**98.0**	**97.0**	**98.0**	**100**	**94.0**
GPT4.1	**94.0**	90.0	87.0	97.0	**97.0**	97.0	96.0	**94.0**
Average (Frontier)	85.4	88.0	85.6	93.6	92.6	92.8	94.6	90.4
Gemma3-12B-IT	34.0	78.0	81.0	66.0	81.0	81.0	81.0	71.7
DeepSeek-R1-Distill-Qwen32B	81.0	69.0	74.0	81.0	74.0	72.0	78.0	75.6
DeepSeek-R1-Distill-Llama70B	85.0	66.0	77.0	83.0	85.0	80.0	87.0	80.4
Llama3.3-70B	66.0	74.0	78.0	94.0	93.0	91.0	98.0	84.9
Gemma3-27B-IT	82.0	75.0	80.0	93.0	93.0	84.0	**99.0**	86.6
Llama3.2-3B	93.0	**97.0**	93.0	69.0	80.0	**99.0**	90.0	88.7
Qwen3-8B	77.0	87.0	93.0	96.0	92.0	91.0	93.0	89.9
Llama3.1-8B	74.0	91.0	**99.0**	91.0	95.0	97.0	97.0	92.0
Qwen3-32B	90.0	94.0	96.0	97.0	92.0	98.0	97.0	94.9
Gemma3-4B-IT	**100**	94.0	92.0	**98.0**	**99.0**	**99.0**	96.0	**96.9**
Avg (Small/Distilled)	78.2	82.5	86.3	86.8	88.4	89.2	91.6	86.1

Table 4.1: Task 1 results: model accuracy on logical-form execution. Columns are ordered by ascending overall average across logic types.

overall capability; (3) the cross-model mean for each logic type to characterize type difficulty.

RQ1: Logical Form Execution Accuracy

Our first research question asks: How well can models execute a gold LF against an input table? We answer this in two stages, both relying on a trace-level comparison between the model-generated trace and the dataset's reference execution: (1) **Per-operation accuracy:** We compare the model-generated logic trace against the gold trace, marking the final result as correct only if the execution paths align completely. As shown in Table 4.1, performance is highly operation-dependent. Notably, even frontier models with an average accuracy exceeding 90%, such as Llama3.1-405B, exhibit drops, sometimes falling to around 80% on *Aggregation*. Overall, frontier-scale models achieve stronger execution accuracy with 90.4% compared with 86.1% in small/distilled models; however, performance

is not uniform across logic types. In particular, both groups exhibit their weakest results on *Aggregation* (Frontier: 85.4% and Small: 78.2%), then *Count* in Small/distilled (82.5%) and *Ordinal* in Frontier models (85.6%), suggesting that multi-step numerical computation and ranking/order-sensitive operators remain fragile even for high-capacity models. Additionally, the logic-specific errors in Table 4.2 underscore the prevalence of computational errors in these specific cases.

In contrast, *Majority*, *Superlative*, and *Only* tend to be more reliable across both tiers (Frontier: 94.6%, 92.8%, 92.6%; Small/Distilled: 91.6%, 89.2%, 88.4%), consistent with these operators often reducing to simpler combinations of filtering and counting/extrema. Notably, scaling benefits are not evenly distributed: the Frontier-Small gap is largest for *Aggregation* and *Comparative*, 7 points for each, while it is smaller for *Majority* and *Superlative*.

(2) **Failure-pattern taxonomy:** To move beyond binary metrics, we leverage the tree-structured nature of the execution traces, which capture operators, arguments, and intermediate row/column subsets, to perform fine-grained mismatch attribution. Through manual inspection of DeepSeek-v3.2 mismatches and LLM-assisted summarization for other frontier models, we clustered these errors into the taxonomy presented in Table 4.2, and more details are given in tables A.2, A.3, and A.4 in the Appendix.

Across all evaluated models, failures separate into cross-cutting and logic-specific categories. Cross-cutting errors, such as primarily incorrect row filtering and inconsistencies in fuzzy versus strict string matching, occur across most operations and constitute the majority of the error mass. In contrast, logic-specific errors stem from operator-sensitive pitfalls, such as miscalculating averages during Aggregation. This taxonomy clarifies why executions fail and motivates straightforward mitigation, including value normalization, tolerant numeric comparison,

Logic type	Execution failure pattern (frontier models)
Aggregation	Average mis-calculation (wrong mean or precision) Sum mis-calculation Rounding/threshold tolerance (e.g., 21.57 vs 21.43) Row-filtering issue (extra or missing rows before aggregation)
Comparative	Numeric-diff mismatch (diff correct but format/string off) Unsupported operation/format (date arithmetic, mixed units) Unit-token mismatch ("6 points" vs "6") Composite-logic failure (sub-clause in an AND/OR chain flips the whole result)
Count	Under-count (qualifying rows missed) Over-count (extra rows counted) Row-filtering issue (substring filters too broad/narrow) Parser/format mismatch (numbers with commas, "15 July" vs "July 15")
Majority	Fuzzy vs. strict equality (`most_eq` vs `most_str_eq`) Off-by-one majority threshold (predicate true, model says false) Comparator-swap / argument-swap ("home" vs "away")
Ordinal	Fuzzy vs. strict equality (`eq` vs `str_eq`) Wrong ordinal index / tie handling (`nth`, `argmin`/`argmax`) Parser/format mismatch
Superlative	Fuzzy vs. strict equality Row-filtering issue (pre-filter too loose)
Only	Row-filtering issue (returns 0 or >1 rows) Fuzzy vs. strict equality

Table 4.2: Most common LF execution failure patterns aggregated across frontier models, grouped by logic type.

and explicit operator disambiguation.

RQ2: LF-to-Text Realization Fidelity (Task 2)

Our second research question investigates: How well can LLMs generate sentences that are both logically entailed by the table and faithful to the provided LF? To investigate this, first, we apply automatic logical fidelity metrics to address the former (Tables 4.3-4.4). In these tables, the section background colors distinguish evaluators (NLI-Acc, TAPAS-Acc, TAPEX-Acc). Bold indicates the strongest

score within each comparison block. Red highlights the weakest average logic type. Green and tan row shading emphasize the selected base model vs. the distilled version in the comparison.

Because these metrics primarily capture entailment rather than strict LF compliance, we additionally conduct a human LF-adherence evaluation to assess whether models truly realize the intended LF (Table 4.6). We present the automatic results first, followed by the human findings.

1) Overall trends across evaluators

The three verifiers yield notably different absolute accuracies, indicating that the perceived difficulty of a logic type can depend strongly on the evaluation backend.

In particular, **NLI-Acc is the harshest on Aggregation**: Aggregation scores are typically in the $\sim$ 40%–60% range, whereas TAPAS-Acc and TAPEX-Acc assign substantially higher Aggregation accuracies (often $\sim$ 70%–90%). This gap suggests that evaluator choice can materially change how hard Aggregation appears. Overall, TAPEX-Acc is the most optimistic (many columns concentrated around 77%–90%), NLI-Acc sits in-between, and TAPAS-Acc is more conservative, especially on count-based structural logic. We therefore emphasize that **absolute accuracies depend on the verifier**, while the **relative difficulty ordering across logic types remains largely stable**.

2) Logic Type Difficulty: Task 2 performance varies sharply across logic types. Although absolute accuracies vary across verifiers, the difficulty ordering across logic types is largely stable.

Model	Aggregation	Comparative	Count	Majority	Ordinal	Superlative	Only	Average
NLI-Acc								
Llama3.1-405B	59.0	83.0	76.0	80.0	49.0	90.0	86.0	75.0
Qwen3-235B-A22B	60.0	80.0	75.0	78.0	43.0	**96.0**	92.0	75.0
Deepseek-v3.2	62.0	85.0	68.0	79.0	55.0	90.0	90.0	76.0
GPT5.1	60.0	**86.0**	77.0	76.0	50.0	91.0	**94.0**	76.0
GPT4.1	58.0	**86.0**	**85.0**	81.0	45.0	94.0	91.0	**77.0**
Average (Frontier)	60.0	84.0	76.0	79.0	48.0	92.0	91.0	76.0
Llama3.2-3B	67.0	69.0	55.0	81.0	**68.0**	82.0	53.0	68.0
Gemma3-27B-IT	43.0	73.0	61.0	74.0	52.0	87.0	83.0	68.0
Deepseek-R1-Distill-Qwen32B	43.0	78.0	63.0	77.0	41.0	95.0	87.0	69.0
Deepseek-R1-Distill-Llama-70B	48.0	81.0	63.0	77.0	42.0	92.0	86.0	70.0
Llama3.3-70B	41.0	80.0	80.0	80.0	43.0	91.0	88.0	72.0
Qwen3-8B	49.0	74.0	74.0	82.0	41.0	94.0	87.0	72.0
Gemma3-12B-IT	55.0	72.0	78.0	83.0	46.0	91.0	85.0	73.0
Llama3.1-8B	58.0	85.0	69.0	79.0	64.0	84.0	73.0	73.0
Qwen3-32B	59.0	80.0	73.0	82.0	40.0	**96.0**	90.0	74.0
Gemma3-4B-IT	**69.0**	82.0	79.0	**86.0**	**68.0**	84.0	84.0	**79.0**
Average (Small/Distilled)	53.0	77.0	70.0	80.0	51.0	90.0	82.0	72.0
TAPAS-Acc								
GPT5.1	77.0	87.0	66.0	74.0	40.0	82.0	83.0	73.0
Deepseek-v3.2	79.0	86.0	61.0	76.0	44.0	84.0	87.0	74.0
Qwen3-235B-A22B	81.0	87.0	66.0	72.0	37.0	85.0	**91.0**	74.0
Llama3.1-405B	75.0	87.0	62.0	**78.0**	44.0	89.0	90.0	75.0
GPT4.1	80.0	**88.0**	67.0	75.0	41.0	89.0	89.0	**76.0**
Average (Frontier)	78.0	87.0	64.0	75.0	41.0	86.0	88.0	74.0
Llama3.2-3B	70.0	51.0	70.0	68.0	42.0	68.0	63.0	62.0
Llama3.3-70B	60.0	86.0	61.0	72.0	34.0	84.0	87.0	69.0
Deepseek-R1-Distill-Llama70B	79.0	82.0	53.0	75.0	35.0	78.0	88.0	70.0
Deepseek-R1-Distill-Qwen32B	79.0	79.0	58.0	73.0	37.0	79.0	84.0	70.0
Gemma3-4B-IT	81.0	72.0	**77.0**	72.0	53.0	75.0	76.0	72.0
Llama3.1-8b	**82.0**	77.0	65.0	69.0	46.0	90.0	76.0	72.0
Gemma3-12B-IT	80.0	78.0	69.0	76.0	46.0	79.0	81.0	73.0
Gemma3-27B-IT	74.0	82.0	66.0	68.0	**56.0**	79.0	85.0	73.0
Qwen3-8B	79.0	84.0	65.0	76.0	39.0	88.0	86.0	74.0
Qwen3-32B	81.0	84.0	**77.0**	71.0	42.0	**93.0**	86.0	**76.0**
Average (Small/Distilled)	77.0	78.0	66.0	72.0	43.0	81.0	81.0	71.0
TAPEX-Acc								
GPT5.1	73.0	85.0	76.0	74.0	50.0	81.0	83.0	75.0
Deepseek-v3.2	85.0	93.0	77.0	80.0	53.0	84.0	86.0	80.0
Qwen3-235B-A22B	86.0	**94.0**	81.0	73.0	50.0	84.0	**90.0**	80.0
Llama3.1-405B	87.0	87.0	**85.0**	80.0	57.0	88.0	88.0	**82.0**
GPT4.1	85.0	92.0	83.0	**82.0**	54.0	88.0	87.0	**82.0**
Average (Frontier)	83.0	90.0	80.0	78.0	53.0	85.0	87.0	79.0
Llama3.2-3B	74.0	48.0	64.0	65.0	45.0	71.0	55.0	60.0
Deepseek-R1-Distill-Qwen32B	93.0	81.0	64.0	76.0	47.0	74.0	84.0	74.0
Gemma3-4B-IT	84.0	73.0	78.0	72.0	**58.0**	78.0	74.0	74.0
Llama3.3-70B	68.0	83.0	70.0	78.0	49.0	84.0	**90.0**	75.0
Deepseek-R1-Distill-Llama70B	93.0	85.0	63.0	83.0	44.0	75.0	86.0	76.0
Gemma3-27B-IT	80.0	87.0	70.0	80.0	55.0	79.0	86.0	77.0
Llama3.1-8B	90.0	83.0	76.0	78.0	49.0	**94.0**	73.0	78.0
Qwen3-8B	88.0	87.0	74.0	79.0	51.0	89.0	86.0	79.0
Qwen3-32B	**93.0**	78.0	77.0	81.0	52.0	90.0	84.0	79.0
Gemma3-12B-IT	93.0	89.0	80.0	81.0	57.0	78.0	81.0	80.0
Average (Small/Distilled)	87.0	79.0	72.0	77.0	51.0	81.0	80.0	75.0

Table 4.3: Automated logical fidelity evaluation of generated sentences on Logic2Text with given LFs.

Model	Aggregation	Comparative	Count	Majority	Ordinal	Superlative	Only	Average
				SP-Acc				
Deepseek-v3.2	24.0	48.0	59.0	67.0	59.0	**83.0**	69.0	58.0
Llama3.1-405B	25.0	41.0	54.0	66.0	67.0	81.0	81.0	59.0
Qwen3-235B-A22B	18.0	42.0	62.0	69.0	66.0	80.0	73.0	59.0
GPT4.1	24.0	50.0	61.0	65.0	64.0	81.0	81.0	61.0
GPT5.1	44.0	**57.0**	61.0	**76.0**	**74.0**	81.0	77.0	**67.0**
Average (Frontier)	27.0	48.0	59.0	69.0	66.0	81.0	76.0	61.0
Llama3.2-3B	20.0	44.0	37.0	57.0	48.0	62.0	69.0	48.0
Gemma3-4B-IT	24.0	55.0	52.0	54.0	65.0	71.0	72.0	56.0
Gemma3-12B-IT	22.0	39.0	57.0	63.0	64.0	75.0	74.0	56.0
Llama3.1-8B	**32.0**	44.0	52.0	65.0	58.0	68.0	80.0	57.0
Qwen3-32B	30.0	41.0	57.0	68.0	57.0	78.0	73.0	58.0
Gemma3-27B-IT	30.0	43.0	57.0	55.0	65.0	**83.0**	76.0	58.0
Deepseek-R1-Distill-Llama70B	29.0	42.0	58.0	70.0	58.0	76.0	81.0	59.0
Qwen3-8B	30.0	42.0	53.0	63.0	68.0	82.0	79.0	60.0
Llama3.3-70B	41.0	41.0	60.0	64.0	59.0	79.0	80.0	60.0
Deepseek-R1-Distill-Qwen32B	29.0	49.0	**65.0**	73.0	59.0	78.0	**84.0**	62.0
Average (Small/Distilled)	29.0	44.0	55.0	63.0	60.0	75.0	77.0	56.0

Table 4.4: Automated logical fidelity evaluation of generated sentences on Logic2Text with given LFs.

Ordinal is consistently the hardest type. Despite differences in scale, all three verifiers agree on the main difficulty profile. Frontier models, the block-average Ordinal accuracies are 48% (NLI), 41% (TAPAS), and 53% (TAPEX); for Small/Distilled models, they are 51%, 43%, and 51%, respectively.

This consistency makes Ordinal the most reliable hard case for qualitative error analysis. Based on these metrics and the failure patterns in Table 4.2, we conclude that ranking and ordering constraints remain fragile even when the underlying program is provided. Ordinal realization requires precise expression of rank position, such as second-highest, and is prone to off-by-one errors, tie ambiguity, and dropped ordinal constraints.

Superlative and Only are the easiest types. Across the same evaluators, these logic types achieve the highest accuracies. These operations map naturally to common lexical patterns such as highest, lowest, and only, making LF

realization comparatively easier.

Comparative, Majority, and Count occupy a middle tier. *Comparative* is generally strong under table-aware evaluators, while errors stem from format-sensitive issues such as numeric–string mismatches or unit inconsistencies. *Majority* is often moderate to strong but susceptible to fuzzy vs. strict equality, quantifier realization, and scope errors. *Count* typically performs in the middle, with common errors arising from counting the wrong subset due to missing or incorrectly realized filters.

SP-Acc highlights Aggregation-specific brittleness. This aligns with the inherent challenges of semantic parsing under linguistic variation: Aggregation statements permit diverse paraphrases, such as *total*, *sum*, *combined*, *difference*, or *average*, where minor mis-parsing of a rounding threshold or a row-filter can flip the execution outcome even when the generated sentence remains table-consistent.

3) Macro-Average Comparison Reveals Verifier-Dependent Winners: The identity of the top-performing model depends on the verifier, underscoring that rankings are not fully stable across evaluation metrics. Under **NLI-Acc**, the best overall average is achieved by **Gemma3-4B-IT** (79%), exceeding all Frontier-model averages; the best Frontier model is **GPT4.1** (77%). This highlights a noteworthy case where a **smaller model can outperform frontier models under an NLI-style verifier**. This suggests that targeted *instruction tuning* can deliver high logical fidelity even at low parameter counts.

Under **TAPAS-Acc**, the best overall average is 76% obtained by **GPT4.1** and **Qwen3-32B**, and Small/Distilled models show greater dispersion. Under

TAPEX-Acc, the best overall average is 82% in **GPT4.1** and **Llama3.1-405b**. While many models cluster near 75%–80%, **Llama3.2-3b** drops sharply (60%). Taken together, these results suggest that *conclusions should prioritize robust per-logic trends and cross-verifier agreement*, rather than a single best model claim.

4) Cross-Model Mean Comparison of Frontier vs. Small/Distilled Models: Performance gaps are not uniform across evaluators, models, or logic types. Using cross-model group averages, Frontier models lead on *Comparative*, *Only*, and *Superlative*, with margins of roughly 2 to 11 points. In contrast, Small-/Distilled models remain competitive with, and sometimes surpass, Frontier peers on *Aggregation* under TAPAS/TAPEX-Acc, and on *Ordinal* under NLI-Acc and TAPAS-Acc. SP-Acc amplifies nearly all gaps, especially for *Aggregation*, where strict parsing penalizes smaller checkpoints more sharply. ***Distilled variants do not yield consistent gains***. For example, the distilled Qwen-32B model matches its base counterpart on *Aggregation* (93%) but drops by 3 to 13 points on *Comparative*, *Count*, and *Majority*, ultimately lowering its macro-average across all entailment-based metrics. Similarly, while distillation strongly boosts *Aggregation* performance from 68% to 93% for Llama-70B, this variant simultaneously loses points on *Count* and *Superlative*, resulting in only a marginal shift in its overall average.

5) Family-level Observations: Table 4.5 compares the models within each family using their macro-average scores across the four metrics; within each family, the models are ordered by parameter count. Among **GPT** models, GPT4.1 is the strongest all-rounder on the entailment-based evaluators, whereas GPT5.1

Family	Model	NLI → TAPAS → TAPEX → SP
GPT	4.1	77 → 76 → 82 → 61
	5.1	76 → 73 → 75 → 67
Llama3	405B	75 → 75 → 82 → 59
	70B	72 → 69 → 75 → 60
	r1-distill-llama-70B (Deepseek)	70 → 70 → 76 → 59
	8B	72 → 72 → 78 → 57
Qwen3	235B	75 → 74 → 80 → 59
	32B	74 → 76 → 79 → 58
	R1-Distill-Qwen32B (Deepseek)	69 → 70 → 74 → 62
	8B	72 → 74 → 79 → 60
DeepSeek	v3.2	76 → 74 → 80 → 58
Gemma3	27B-IT	68 → 73 → 77 → 58
	12B-IT	73 → 73 → 80 → 56
	4B-IT	79 → 72 → 74 → 56

Table 4.5: Family comparison using macro-average scores across evaluator metrics.

gives up that lead but crashes far less under the strict SP-Acc setting, outperforming 4.1 by 8 points. In **Llama3**, the 405B model is the clear winner across evaluators, the 8B model is surprisingly strong for its size, and the 70B model sits between them, although its distilled counterpart performs better on SP-Acc. In **Qwen3**, the 235B model achieves the strongest strict score and dominates harder logic cases, but the 32B model remains very close; specifically, 32B delivers roughly 95% of the fidelity at about 15% of the size, suggesting a strong efficiency-performance tradeoff. In **Gemma3**, the 4B-IT model is best overall and trails by only three points on the strict parser. The 4B-IT model is therefore a budget star for arithmetic tasks.

Key observations:

- **Larger model is not always better within a family.** Llama3-8B surpasses its 70-B sibling on two of the four evaluators, Qwen3-32B keeps pace with the 235-B flagship, and the compact Gemma3-4B-IT outshines its family's largest model on most metrics.

- **Strict-parser robustness can invert the ranking.** GPT5.1 overtakes GPT4.1 only under SP-Acc; Pick GPT4.1 for general-purpose fidelity; GPT5.1 if your pipeline must survive strict program recovery.

- **Specialization appears even within the same family.** GPT models perform well in *Comparative*, Qwen models are especially strong on *Only* and *Superlative*; Gemma models on *Aggregation* and *Majority*.

6) Human LF-Adherence Evaluation: To probe whether Task 2 outputs faithfully realize the provided LFs, we conduct a focused human study on *Gemma3-27B-IT*. This model sits near the overall mean performance of both the *Frontier* and *Small/Distilled* groups under the more closely aligned table-based verifiers (TAPAS/TAPEX-Acc).

For each logic type, we annotate 40 model outputs sampled from three strata whenever possible:

1. *Agreement-Entailed (E/E)*: 15 instances where both verifiers predict entailed,

2. *Agreement-Refuted (R/R)*: 15 instances where both verifiers predict not-entailed/refused

3. *Disagreement*: 10 instances where TAPAS and TAPEX disagree.

Logic	$\textbf{Entailed}_{both}$	$\textbf{Refuted}_{both}$	$\textbf{Refuted}_{TAPAS}$	$\textbf{Refuted}_{TAPEX}$	A	N	O	U
Majority	15	13	6	6	40	0	0	0
Aggregation	15	15	7	3	36	3	1	0
Comparative	16	7	11	6	36	2	0	2
Only	17	6	9	8	35	5	0	0
Ordinal	15	15	5	5	32	5	2	1
Superlative	16	16	4	4	29	10	1	0
Count	15	15	5	5	20	16	3	1

Table 4.6: LF-adherence investigation by a human study.

This stratified design allows us to measure LF-adherence while diagnosing whether verifier outcomes reflect true generation errors or evaluator noise.

Each instance includes the input table, the model-generated claim (Task 2 output), the reference LF, and its natural-language interpretation. A graduate student annotator voluntarily participated and provided informed consent before labeling the samples. The evaluator judged LF-adherence of each instance by answering: *Does the generated claim faithfully realize the provided logical form when grounded in the table?* The annotator selects one of:

- *Adheres (A):* The sentence realizes the logical form completely and correctly.
- *Not-entailed (N):* The sentence cannot be verified as true from the table.
- *Off-LF but entailed (O):* The sentence is supported by the table, but expresses a different fact than the LF, such as a different operator, column, subset/filter, or missing/extra constraint.
- *Unclear (U):* The sentence is too vague or underspecified to verify.

Table 4.6 summarizes the results. ***High LF-adherence overall.*** Five of seven logic types exhibit almost 90% adherence; the exceptions are *Count* (50%)

and *Superlative* (72%). ***Verifier refutations often hide false negatives.*** In the *refuted-by-both* subset, almost 45% of cases are still rated **A** by human (73% for Aggregation, 100% for Majority, and at least 47% for Ordinal computed using Equation4.1). ***Dual acceptance is highly reliable.*** When both verifiers accept a claim, this indicates that combined acceptance is a strong indicator of the LF faithfulness.

We introduce the *Verifier False Negative Rate (VFNR)* to quantify how often verifiers jointly reject claims that a human still deems LF-faithful. The value is computed in Equation 4.1:

$$\text{VFNR} = \frac{A - \text{Entailed}_{\text{both}} - \text{Refuted}_{\text{TAPAS}} - \text{Refuted}_{\text{TAPEX}}}{\text{Refuted}_{\text{both}}} \tag{4.1}$$

Where A is the total number of *Adheres* approved by a human evaluator. $\text{Refuted}_{\text{both}}$: The total number of cases rejected by both automatic verifiers. $\text{Refuted}_{\text{TAPAS}}$: The total number of cases rejected by TAPAS. $\text{Refuted}_{\text{TAPEX}}$: The total number of cases rejected by TAPEX.

We conclude that a high VFNR indicates that dual-metric rejection is overly aggressive, hiding many genuine, human-approved outputs.

Task 2 Takeaways. Task 2 confirms that LF-conditioned generation remains challenging and operation-dependent. While automatic verifiers (NLI/-TAPAS/TAPEX) provide a useful entailment baseline, human inspection reveals that many *refutations* stem from evaluator limitations rather than model errors. Conversely, when verifiers agree on entailment, the outputs are consistently LF-faithful. Single aggregate scores thus conflate (1) genuine realization errors, (2) Off-LF yet entailed outputs, and (3) verifier false rejections, underscoring the need for more nuanced, operation-aware evaluation.

Model	Ordinal	Superlative	Aggregation	Count Only	Majority	Comparative	Overall
GPT5.1	29.0	49.0	38.0	35.0	69.0	100	100	60.0
Qwen3-235B-A22B	53.0	67.0	56.0	70.0	40.0	99.0	86.0	67.3
Llama3.1-405B	17.0	38.0	70.0	100	95.0	97.0	100	73.9
GPT4.1	48.0	93.0	86.0	98.0	99.0	99.0	96.0	88.0
Deepseek-v3.2	93.0	93.0	98.0	77.0	95.0	95.0	97.0	93.0
Average	48.0	68.0	69.6	76.0	79.6	98.0	96.0	76.0
Llama3.2-3B	17.0	15.0	3.0	16.0	53.0	8.0	91.0	29.0
Gemma3-4B-IT	1.0	0.0	95.0	9.0	23.0	4.0	81.0	30.0
Gemma3-27-IT	0.0	0.0	5.0	11.0	33.0	100	100	36.0
Llama3.1-8B	0.0	0.0	0.0	92.0	80.0	74.0	98.0	49.0
Qwen3-8B	0.0	26.0	0.0	44.0	98.0	84.0	98.0	50.0
Llama3.3-70B	1.0	22.0	11.0	49.0	76.0	100	100	51.0
Gemma3-12B-IT	18.0	45.0	18.0	41.0	39.0	99.0	100	51.0
Qwen3-32B	21.0	45.0	38.0	45.0	91.0	100	96.0	62.0
Deepseek-R1-Distill-Llama70B	76.0	85.0	74.0	84.0	60.0	100	100	83.0
Deepseek-R1-Distill-Qwen32B	71.0	80.0	68.0	79.0	59.0	100	100	80.0
Average	21.0	32.0	31.0	47.0	61.0	77.0	96.0	52.0

Table 4.7: Accuracy of models in logic type prediction

RQ3: Logic-Type Identification (Task 3)

Our third research question checks whether models can predict the logic type of the generated sentence correctly. Table 4.7 reports per-type and overall accuracy for logic-type prediction. Overall, most models achieve high accuracy on *Comparative* and *Majority* (except the smallest models), but struggle on *Ordinal*, *Aggregation*, and *Superlative*.

Figure 4.2 reveals the systematic confusions of models in logic type prediction. In particular, *Comparative* is over-predicted: instances from *Aggregation*, *Ordinal*, and *Superlative* are frequently misclassified as *Comparative*. A closer inspection of model rationales reveals a systematic error: many models focus on the *final boolean check* in the LF—which indeed compares a derived value to the gold reference—while ignoring the *primary* operation used to *produce* that value. For example, consider the instance

> **Sentence generated by the model:** "The average race rank is approximately 12.56."

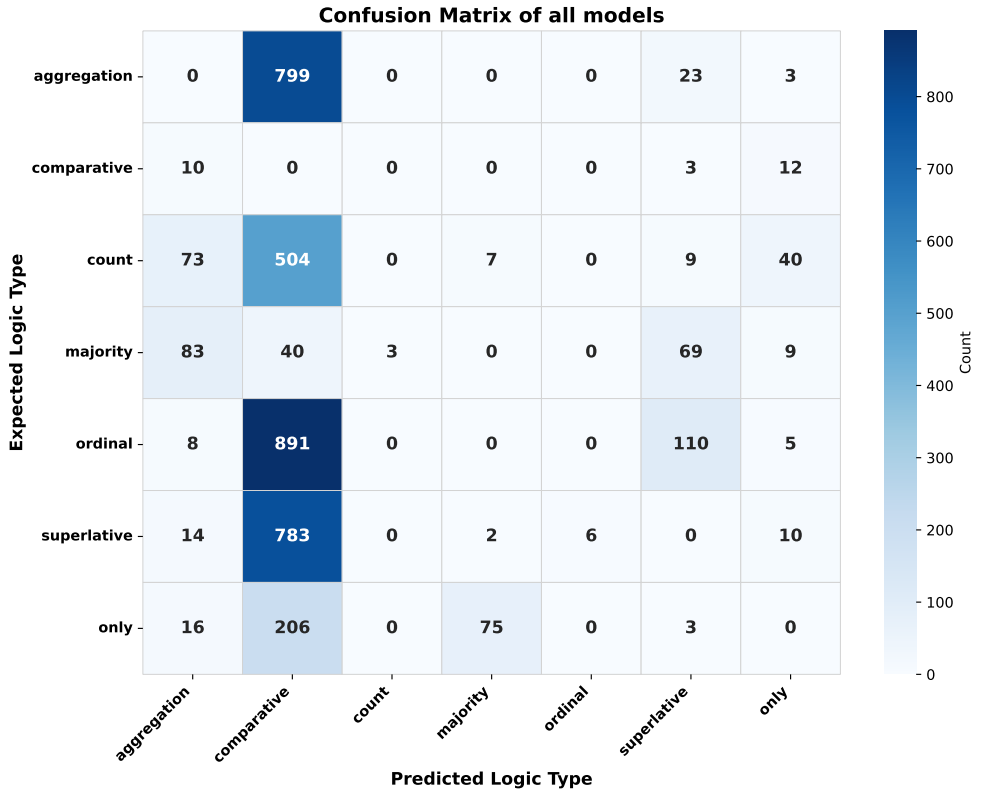


Figure 4.2: Confusion matrix heatmap of all models.

Logic string: `round_eq { avg { all_rows ; rank } ; 12.56 } = true`

The correct logic type is *Aggregation* because the central action is computing an *average* over all rows. However, several models label it as *Comparative*, explaining that "the logic string checks if the average of the *rank* column is approximately equal to 12.56." In effect, the model attends to the `round_eq` comparison at the end of the LF rather than the `avg` operation that dominates the reasoning chain.

This *last-step bias* appears across diverse architectures and accounts for a substantial fraction of false *Comparative* predictions. Accordingly, we believe that the over-prediction of *Comparative* is not an inherent weakness of current

LLMs, but a resolvable prompting artifact, and they motivate future work on prompt engineering and rationale supervision for logic-type classification.

RQ4: LF-Free Same-Type Claim Generation (Task 4)

Task 4 evaluates whether models can generate a new entailed claim *without* access to an LF. Tables 4.8 and 4.9 report sentence-level correctness for Task 4 using the same automatic fidelity metrics. The same color codes as Task 2 are used in these tables.

Impact of Removing Logical Forms. Tables 4.3–4.4 (NLI/TAPAS/TAPEX)-Acc and Tables 4.8–4.9 (SP-Acc) contrast logical-fidelity scores *with* versus *without* the gold LF. ***Removing the LF consistently degrades performance***. The effect is most pronounced for TAPAS/TAPEX and for smaller or distilled models.

Frontier models. Macro-average TAPAS/TAPEX-Acc falls from 74% to 69% and from 79% to 73%, respectively. SP-Acc drops from 61% to 56%.

Small/Distilled models. The decline is sharper: TAPAS/TAPEX-Acc fall 10 and 11 points, respectively, while SP-Acc decreases 4 points.

Model	Aggregation	Comparative	Count	Majority	Ordinal	Superlative	Only	Average
NLI-Acc								
Deepseek-v3.2	68.0	82.0	64.0	68.0	56.0	88.0	79.0	72.0
Llama3.1-405B	57.0	81.0	82.0	64.0	81.0	91.0	70.0	75.0
GPT5.1	61.0	86.0	81.0	66.0	73.0	85.0	85.0	77.0
Qwen3-235B-A22B	72.0	83.0	79.0	67.0	76.0	89.0	83.0	78.0
GPT4.1	71.0	88.0	85.0	68.0	68.0	95.0	79.0	79.0
Average (Frontier)	65.8	84.0	78.2	66.6	70.8	89.6	79.2	76.2
Llama3.2-3B	72.0	66.0	60.0	73.0	60.0	73.0	60.0	66.0
Gemma3-27B-IT	47.0	74.0	65.0	57.0	77.0	71.0	70.0	66.0
Llama3.1-8B	47.0	80.0	71.0	68.0	58.0	82.0	65.0	67.0
Deepseek-R1-Distill-Qwen32B	38.0	73.0	60.0	73.0	65.0	86.0	84.0	68.0
Deepseek-R1-Distill-Llama70B	43.0	78.0	66.0	67.0	74.0	82.0	77.0	69.0
Gemma3-12B-IT	65.0	75.0	72.0	69.0	68.0	82.0	79.0	73.0
Qwen3-8B	65.0	75.0	72.0	79.0	39.0	91.0	91.0	73.0
Gemma3-4B-IT	68.0	73.0	80.0	68.0	77.0	79.0	78.0	75.0
Qwen3-32B	70.0	79.0	71.0	78.0	53.0	88.0	83.0	75.0
Llama3.3-70B	59.0	79.0	82.0	70.0	85.0	89.0	67.0	76.0
Average (Small/Dist.)	57.4	75.2	69.9	70.2	65.6	82.3	75.4	70.8
TAPAS-Acc								
Deepseek-v3.2	83.0	83.0	58.0	43.0	49.0	75.0	67.0	66.0
GPT5.1	67.0	88.0	66.0	37.0	66.0	79.0	77.0	69.0
Llama3.1-405B	83.0	85.0	63.0	45.0	71.0	82.0	53.0	69.0
Qwen3-235B-A22B	82.0	82.0	61.0	55.0	72.0	77.0	70.0	71.0
GPT4.1	79.0	85.0	66.0	51.0	67.0	82.0	72.0	72.0
Average (Frontier)	78.8	84.6	62.8	46.2	65.0	79.0	67.8	69.4
Llama3.1-8B	44.0	75.0	52.0	29.0	47.0	48.0	51.0	49.0
Llama3.2-3B	52.0	54.0	53.0	48.0	36.0	63.0	58.0	52.0
Deepseek-R1-Distill-Llama70B	72.0	70.0	44.0	44.0	51.0	53.0	64.0	57.0
Deepseek-R1-Distill-Qwen32B	68.0	74.0	54.0	45.0	45.0	52.0	66.0	58.0
Gemma3-27B-IT	61.0	76.0	62.0	52.0	57.0	63.0	54.0	61.0
Gemma3-12B-IT	79.0	72.0	61.0	52.0	58.0	55.0	55.0	62.0
Gemma3-4B-IT	71.0	65.0	67.0	50.0	63.0	62.0	67.0	64.0
Llama3.3-70B	82.0	73.0	69.0	47.0	68.0	75.0	58.0	67.0
Qwen3-8B	87.0	81.0	65.0	60.0	37.0	76.0	80.0	69.0
Qwen3-32B	87.0	75.0	59.0	60.0	52.0	73.0	81.0	70.0
Average (Small/Dist.)	70.3	71.5	58.6	48.7	51.4	62.0	63.4	60.9
TAPEX-Acc								
Deepseek-v3.2	90.0	79.0	66.0	49.0	54.0	80.0	66.0	69.0
Llama3.1-405B	89.0	85.0	83.0	49.0	76.0	79.0	44.0	72.0
GPT5.1	69.0	87.0	81.0	49.0	74.0	80.0	81.0	74.0
GPT4.1	82.0	89.0	84.0	44.0	76.0	83.0	65.0	75.0
Qwen3-235B-A22B	93.0	79.0	75.0	61.0	70.0	83.0	62.0	75.0
Average (Frontier)	84.6	83.8	77.8	50.4	70.0	81.0	63.6	73.0
Llama3.1-8B	39.0	81.0	50.0	29.0	50.0	51.0	44.0	49.0
Llama3.2-3B	61.0	51.0	59.0	51.0	41.0	64.0	46.0	53.0
Deepseek-R1-Distill-Llama70B	91.0	64.0	60.0	41.0	58.0	62.0	51.0	61.0
Deepseek-R1-Distill-Qwen32B	95.0	67.0	65.0	40.0	49.0	58.0	53.0	61.0
Gemma3-4B-IT	79.0	61.0	64.0	54.0	62.0	66.0	68.0	65.0
Gemma3-27B-IT	77.0	77.0	70.0	54.0	67.0	73.0	50.0	67.0
Gemma3-12B-IT	90.0	74.0	73.0	56.0	65.0	66.0	53.0	68.0
Llama3.3-70B	87.0	74.0	74.0	46.0	79.0	73.0	54.0	70.0
Qwen3-8B	92.0	78.0	68.0	62.0	55.0	77.0	74.0	72.0
Qwen3-32B	91.0	68.0	69.0	70.0	56.0	80.0	85.0	74.0
Average (Small/Dist.)	80.2	69.5	65.2	50.3	58.2	67.0	57.8	64.0

Table 4.8: Task 4: automated logical-fidelity evaluation on Logic2Text *without* provided LFs (higher is better).

Model	Aggregation	Comparative	Count	Majority	Ordinal	Superlative	Only	Average
				SP-Acc				
Llama3.1-405B	16.0	41.0	53.0	67.0	65.0	59.0	57.0	51.0
Deepseek-v3.2	20.0	51.0	61.0	70.0	60.0	64.0	61.0	55.0
GPT4.1	22.0	52.0	52.0	72.0	74.0	64.0	75.0	59.0
Qwen-3-235B	25.0	45.0	60.0	73.0	71.0	72.0	73.0	60.0
GPT5.1	36.0	54.0	51.0	76.0	71.0	58.0	75.0	60.0
Average (Frontier)	23.8	48.6	55.4	71.6	68.2	63.4	68.2	57.0
Llama3.2-3B	33.0	40.0	44.0	58.0	48.0	54.0	46.0	46.0
Gemma3-12B-IT	22.0	35.0	63.0	71.0	58.0	39.0	61.0	50.0
Gemma3-27B-IT	28.0	44.0	49.0	69.0	62.0	49.0	53.0	51.0
Deepseek-R1-Distill-Llama70B	15.0	44.0	47.0	69.0	66.0	55.0	65.0	51.0
Llama3.3-70B	20.0	43.0	56.0	76.0	68.0	49.0	55.0	52.0
Deepseek-R1-Distill-Qwen32B	11.0	53.0	50.0	66.0	64.0	49.0	76.0	53.0
Gemma3-4B-IT	32.0	50.0	51.0	56.0	70.0	56.0	56.0	53.0
Llama3.1-8B	57.0	46.0	36.0	62.0	53.0	52.0	64.0	53.0
Qwen-3-8B	23.0	51.0	43.0	64.0	65.0	54.0	68.0	53.0
Qwen-3-32B	29.0	48.0	52.0	69.0	62.0	66.0	76.0	57.0
Average (Small/Dist.)	27.0	45.4	48.9	66.0	61.6	52.3	62.0	51.9

Table 4.9: Task 4: SP-Acc on Logic2Text *without* LFs .

Which logic types are most affected. The LF yields the largest gains for operations that impose explicit constraints, notably *Majority*, *Superlative*, and *Only*. Without the LF, *Majority* accuracy collapses for frontier models (TAPAS: 75% to 46%; TAPEX: 78% to 50%), implying that unconstrained models struggle to preserve majority-style evidence requirements. Under SP-Acc, the steepest drops occur for *Superlative* (frontier: –19 pts; small/distilled: –23 pts) and *Only* (frontier: –9 pts; small/distilled: –15 pts). Interestingly, *Ordinal* sometimes *improves* when the LF is removed: for frontier models, TAPAS rises by 24 pts and TAPEX by 17 pts, consistent with models producing easier-but-entailed statements when not required to follow the target operation.

Overall, LFs provide strong operator-level guidance, with the greatest impact on capacity-limited models and on operations requiring non-trivial constraints.

Logic type	NLI–TAPAS	NLI–TAPEX	TAPAS–TAPEX
Ordinal	0.54	**0.61**	0.38
Majority	0.58	**0.62**	0.37
Count	0.70	**0.73**	0.59
Aggregation	0.61	**0.75**	0.61
Only	0.70	**0.76**	0.51
Comparative	0.74	**0.79**	0.58
Superlative	0.79	**0.82**	0.48
Overall Avg	0.67	**0.73**	0.50

Table 4.10: Average pairwise disagreement rates across logic types (over models).

4.5 Discussion

In the following subsections, we first discuss the disagreements among metrics and then their reliability in more detail.

4.5.1 Metric Disagreement Analysis.

So far, we notice obvious disagreements among metrics. To evaluate the disagreement rate among metrics, we identify, for each entailment-based verifier, those samples that are flagged as *not entailed* (i.e., *refused*), and then compute pairwise overlaps between these refusal sets to obtain *disagreement rates*. Table 4.10 shows the average results, and Table 4.11 reports the detailed results of pairwise disagreements for each logic type. The main takeaways are:

NLI-Acc diverges most from table-based verifiers. Across logic types, NLI-Acc shows substantially higher disagreement with TAPEX-Acc (avg 0.73) and TAPAS-Acc (avg 0.67) than TAPAS-Acc and TAPEX-Acc disagree with each other (avg 0.50). This gap suggests that NLI models operate on a fundamentally different definition of entailment than dedicated table-based verifiers.

model	Ordinal	Majority	Aggregation	Count	Only	Comparative	Superlative	Average
Pairwise disagreement rate between NLI-Acc and TAPEX-Acc								
Gemma3-27B-IT	0.60	0.49	0.59	0.65	0.75	0.80	0.76	0.66
Llama3.1-8B	0.56	0.67	0.47	0.75	0.68	0.78	0.88	0.68
Llama3.2-3B	0.66	0.71	0.81	0.63	0.59	0.66	0.76	0.69
Deepseek-R1-Distill-Qwen32B	0.53	0.69	0.92	0.59	0.83	0.68	0.71	0.71
Deepseek-R1-Distill-Llama70B	0.67	0.62	0.88	0.66	0.74	0.78	0.65	0.71
Gemma3-12B-IT	0.57	0.61	0.82	0.72	0.83	0.72	0.73	0.71
GPT4.1	0.56	0.58	0.62	0.76	0.81	0.79	0.84	0.71
Qwen3-32B	0.51	0.69	0.88	0.71	0.75	0.74	0.72	0.71
Deepseek-v3.2	0.39	0.62	0.83	0.69	0.78	0.82	0.90	0.72
Qwen3-235B-A22B	0.54	0.43	0.79	0.82	0.75	0.85	0.83	0.72
Gemma3-4B-IT	0.78	0.65	0.69	0.72	0.74	0.69	0.88	0.74
Qwen3-8b	0.42	0.77	0.87	0.79	0.87	0.84	0.77	0.76
GPT5.1	0.71	0.69	0.51	0.77	0.90	0.96	0.91	0.78
Llama3.1-405B	0.74	0.57	0.88	0.79	0.66	0.94	0.93	0.79
Llama3.3-70B	0.94	0.55	0.71	0.84	0.75	0.82	0.97	0.80
Average	0.61	0.62	0.75	0.73	0.76	0.79	0.82	0.73
Pairwise disagreement rate between NLI-Acc and TAPAS-Acc								
Gemma3-27B-IT	0.60	0.43	0.41	0.60	0.73	0.81	0.71	0.61
Llama3.2-3B	0.57	0.70	0.60	0.68	0.50	0.67	0.72	0.63
Deepseek-R1-Distill-Qwen32B	0.53	0.63	0.63	0.49	0.81	0.66	0.75	0.64
Llama3.1-8B	0.52	0.64	0.40	0.72	0.71	0.64	0.83	0.64
Deepseek-v3.2	0.35	0.52	0.68	0.67	0.71	0.70	0.88	0.64
Qwen3-32B	0.33	0.60	0.70	0.72	0.74	0.76	0.66	0.64
Qwen3-235B-A22B	0.47	0.47	0.65	0.67	0.66	0.75	0.83	0.64
Deepseek-R1-Distill-Llama70B	0.62	0.55	0.66	0.63	0.73	0.75	0.62	0.65
Qwen3-8B	0.26	0.67	0.70	0.70	0.83	0.71	0.78	0.66
GPT4.1	0.59	0.50	0.61	0.83	0.60	0.88	0.79	0.69
GPT5.1	0.51	0.61	0.50	0.80	0.81	0.87	0.76	0.69
Llama3.1-405B	0.50	0.53	0.72	0.78	0.57	0.79	0.92	0.69
Gemma3-12B-IT	0.65	0.61	0.67	0.80	0.80	0.64	0.79	0.71
Llama3.3-70B	0.82	0.59	0.60	0.71	0.56	0.77	0.97	0.72
Gemma3-4B-IT	0.77	0.67	0.66	0.77	0.74	0.75	0.79	0.73
Average	0.54	0.58	0.61	0.70	0.70	0.74	0.79	0.67
Pairwise disagreement rate between TAPAS-Acc and TAPEX-Acc								
Llama3.1-8B	0.34	0.21	0.31	0.47	0.36	0.48	0.42	0.37
Deepseek-R1-Distill-Llama70B	0.26	0.25	0.78	0.46	0.56	0.42	0.31	0.43
Gemma3-4B-IT	0.30	0.33	0.45	0.60	0.50	0.40	0.44	0.43
Deepseek-R1-Distill-Qwen32B	0.23	0.35	0.84	0.51	0.49	0.50	0.26	0.45
Llama3.2-3B	0.33	0.43	0.62	0.40	0.40	0.45	0.51	0.45
Gemma3-27B-IT	0.42	0.38	0.52	0.52	0.34	0.53	0.51	0.46
Llama3.3-70B	0.53	0.30	0.52	0.61	0.56	0.49	0.56	0.51
Llama3.1-405B	0.44	0.42	0.78	0.62	0.34	0.57	0.44	0.52
Gemma3-12B-IT	0.46	0.49	0.65	0.60	0.47	0.54	0.45	0.52
Deepseek-v3.2	0.25	0.40	0.65	0.60	0.57	0.64	0.64	0.54
Qwen3-235B-A22B	0.39	0.35	0.75	0.67	0.55	0.70	0.40	0.54
GPT5.1	0.50	0.37	0.44	0.71	0.69	0.86	0.59	0.56
Qwen3-8B	0.34	0.45	0.60	0.62	0.58	0.72	0.58	0.56
GPT4.1	0.42	0.33	0.50	0.78	0.63	0.70	0.60	0.57
Qwen3-32B	0.49	0.52	0.76	0.65	0.63	0.64	0.43	0.59
Average	0.38	0.37	0.61	0.59	0.51	0.58	0.48	0.50

Table 4.11: Pairwise disagreement across metric pairs

The significant disagreement between NLI-Acc and the table-centric models indicates that NLI processes logic through a linguistic lens rather than a tabular one. TAPAS and TAPEX show relatively closer agreement, likely because they are architecturally optimized for structured data.

Disagreement is operation-dependent. The largest gaps appear for *superlative* and *Comparative* (NLI-Acc vs. TAPEX-Acc: 0.82/0.79), followed by *Only* and *Aggregation*, while *Majority* and *Ordinal* exhibit the lowest cross-metric disagreement. Therefore, we can conclude that disagreement spikes on logic types that require precise numeric computation, comparison, and selection.

Model-specific trends. Looking at individual models in Table 4.10, we can see which ones cause the most confusion among the evaluators:

Most Contentious: Models like Llama3.3-70B and GPT5.1 often trigger the highest disagreement rates, reaching up to 0.97 in Superlative logic for Llama3.3-70B. This may imply these models produce edge case answers that different evaluators interpret in opposite ways.

Most Consistent: Smaller or older models like Gemma3-27B-IT, and Llama3.1-8B generally show lower disagreement rates across the board compared to the frontier models.

Aggregation anomaly. The data highlights that *Aggregation* is a specific failure point for table-based reasoning. While TAPAS-Acc and TAPEX-Acc generally align (avg 0.50), their disagreement increases to 0.61 for Aggregation tasks. This suggests that even table-specialized verifiers handle Aggregation differently, resulting in greater disagreement.

4.5.2 Qualitative Error Analysis

As we discussed in the human study 4.4.3, there is a high chance of auto metrics classifying a claim as refuted wrongly. Thus, we are interested in knowing how reliable a verifier's refused/not-entailed decision is, i.e., when a verifier flags an instance as refused, how often is that flag actually justified. We construct a small test study by selecting two subsets with the largest disagreement in refused predictions across verifiers. In our experiments, the strongest disagreement occurs between NLI and TAPEX in two settings: (1) Llama3.3-70B on Superlative instances and (2) GPT5.1 on Comparative instances (Table 4.11). In each subset, the two verifiers agree on only one refused instance. NLI-Acc flags 10 (Superlative) and 13 (Comparative) instances as refused, whereas TAPEX-Acc flags 26 and 12, respectively. Table 4.12 reports the resulting refusal precision, the fraction of refused flags that are correct.

Model	Metric	correct refusal	wrong refusal
Llama3.3-70B	NLI-only	2	8
Llama3.3-70B	TAPEX-only	17	9
GPT5.1	NLI-only	3	10
GPT5.1	TAPEX-only	4	8

Table 4.12: Directional analysis on disagreement-sampled claims.

Overall, NLI-Acc produces more incorrect refusals than TAPEX-Acc in both case studies, although the magnitude of the gap depends on the setting. For Llama3.3-70B on Superlatives, NLI-Acc performs poorly, achieving only 20% refusal precision, while TAPEX-Acc reaches 65%. For GPT5.1 on Comparatives, the difference narrows substantially: TAPEX-Acc remains higher, but the gap drops to roughly 8 percentage points. In our small test set, TAPEX-Acc looks more accurate than NLI-Acc.

4.6 Conclusions

We introduced an operation-aware diagnostic framework for Logical Table-to-Text generation that decomposes performance into four competencies: LF execution, LF-conditioned realization, logic-type identification, and LF-free same-type generation. Across a 700-instance Logic2Text-based benchmark spanning seven logic types, our findings show that strong aggregate fidelity can be misleading: verifier choice substantially alters conclusions, and models frequently exhibit a meta-logical gap, producing statements judged faithful while failing to recognize the underlying operation or to reliably execute the corresponding LF. The results highlight persistent bottlenecks in Ordinal and Aggregation reasoning, while the human LF-adherence study further shows that many outputs labeled as "refuted" reflect verifier limitations rather than true model errors. Overall, these findings motivate operation-level reporting and diagnostics as a necessary step toward building LT2T systems that are not only fluent but also logically reliable in the specific reasoning skills required by real applications.

Chapter 5

Conclusions

This dissertation argues that emerging AI systems must not only produce correct outputs but also provide explanations whose reasoning can be verified against structured evidence. Across two problem domains— UAV navigation and logical table-to-text (LT2T) generation— we show that fluent natural language rationales are not sufficient evidence of faithfulness. Instead, reliability depends on whether models can perform *operation-level reasoning* (e.g., aggregation, comparison, majority, ordinal decisions) consistently and transparently, and whether our evaluation methods can detect and diagnose failures at that level.

We begin with *NavChat*, a post-hoc explanation framework for deep reinforcement learning (DRL) UAV navigation policies. NavChat established that large language models (LLMs) can generate post-hoc rationales that improve interpretability and increase human trust in UAV navigation policies. However, our analyses reveal systematic reasoning failures, recurring mismatches between explanation fluency and underlying correctness: explanations may sound plausible while misrepresenting some values, mishandling logic types such as comparison, or mislabeling. These findings motivate the dissertation's central question: *Do*

current models possess reliable operator-level reasoning over structured evidence, and how can we measure and diagnose that reliability?

To pursue this question in a controlled setting, we pivot to LT2T, where tables provide explicit, auditable evidence and where claims can be grounded and checked. First, we contribute a comprehensive LT2T survey that organizes the space around (1) challenges addressed in LT2T studies, (2) a taxonomy of methods that addressed those challenges, and (3) limitations and future directions. The survey consolidates the field's progress while highlighting a key gap: much of the existing metrics for *logical reasoning* is still reported in aggregate scores that obscure which operations succeed, and where models fail.

Building on these insights, we then introduce an *operation-aware diagnostic framework* that decomposes logical reasoning into verifiable steps. Rather than treating entailment as a single score, our approach evaluates models' outputs for different logic types using automated metrics and structured traces, such as logical-form execution traces and verifier outcomes, to pinpoint failure modes; for instance, wrong column selection, wrong aggregation scope, tie-handling errors, missing constraints, and verifier brittleness. This operation-level view enables finer-grained comparisons across models and evaluators, revealing that models can be simultaneously strong on some operators and unreliable on others, and that different automatic metrics may encode distinct, and sometimes conflicting definitions of correctness.

Taken together, the dissertation contributes a coherent methodology for studying reasoning reliability: (1) expose the limits of fluent rationalization in embodied decision-making (NavChat), (2) formalize the operator-level landscape and evaluation gaps in LT2T (survey), and (3) provide operation-aware diagnostics that explain why a system fails and which operations are responsible. This pro-

gression supports a broader conclusion: progress in reasoning should be measured not only by end-task performance, but by how reliably models execute and justify atomic operations over structured evidence, and by whether evaluation tools can faithfully reflect that reliability.

Chapter 6

Future Work

This dissertation demonstrates that fluent natural language is not sufficient evidence of faithful reasoning. Across both UAV navigation explanation (NavChat) and logical table-to-text generation (LT2T), we observed recurring failure modes—operator mistakes, scope errors, and confident hallucinations—that motivate a next phase focused on *reliability*: systems that (1) reason over structured evidence, and (2) expose the reasoning in an auditable form. Below, we outline concrete directions that extend the research thread.

Toward Process-Aware and Explainable Metrics One direction is to make the evaluation process itself more transparent by incorporating process-level judgments rather than relying only on final entailment decisions.

- **Step-wise scoring.** Instead of scoring only whether a final statement is entailed, the metric could evaluate intermediate reasoning steps such as entity selection, row filtering, aggregation, and comparison. This would make evaluation more interpretable and reveal where reasoning begins to fail.

- **Disagreement-aware evaluation.** When verifiers such as TAPAS-Acc, TAPEX-Acc, and NLI-Acc disagree, their disagreement should be treated not merely as noise but as evidence of uncertainty. Modeling these disagreements explicitly may help reveal evaluator-specific blind spots, especially for difficult operator families such as ordinal and aggregation.

- **Diagnostic feedback generation.** Rather than producing only a scalar score, future metrics could return actionable feedback, such as identifying likely scope errors, incorrect subsets, or aggregation over the wrong rows. In this way, the metric would function not only as an evaluator but also as a diagnostic assistant.

Richer Benchmarks. The framework should be broadened to reflect more realistic forms of evidence and uncertainty.

- **Multi-modal tasks.** Evaluate image–table pairs to see whether additional modalities compound or mitigate current operation-level weaknesses.

- **Multi-table and hierarchical evidence**. Real applications often require joins across multiple tables, hierarchical schemas, or semi-structured sources. Extending the framework to these settings would test whether operator-level diagnostics remain effective when evidence is distributed and compositional.

- **Temporal reasoning**. Future work should incorporate temporal operators, such as before/after relations, trends, and rolling aggregates, into the diagnostic taxonomy.

- **Datasets coverage**. The current framework has only been validated on a

single benchmark; evaluating it on a diverse set of public corpora, such as LogicNLG and HiTab, can expose domain or schema biases, and stress-test the breadth of the diagnostic taxonomy.

Summary. Future work should move beyond raw accuracy toward *auditable, operator-level* evaluation. For **NavChat**, this means attaching each explanation to the concrete evidence it cites, verifying that evidence programmatically, and reporting calibrated uncertainty so end-users can see *both* the model's reasoning path and its confidence in that path. For the **Logical Table-to-Text (LT2T)** setting, the focus shifts to scalable claim decomposition and execution-based checks that do *not* rely on gold logical forms; such checks must pinpoint exactly which operator fails and why. Together, these directions advance the dissertation's overarching goal: building and evaluating models whose reasoning over structured evidence is transparent, diagnosable, and therefore continuously improvable.

Bibliography

Yu Bai, Baoqiang Liu, Shuang Xue, Fang Cai, Na Ye, and Guiping Zhang. Reasoning knowledge filter for logical table-to-text generation. In *Proceedings of Bridging Neurons and Symbols for Natural Language Processing and Knowledge Graphs Reasoning @ COLING 2025*, pages 18–30, Abu Dhabi, UAE, January 2025. ELRA and ICCL.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`. NeurIPS 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023. doi: 10.48550/arXiv.2303.12712.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, May 2024. doi: 10.1145/3641289. URL `https://doi.org/10.1145/3641289`.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, 2020a. doi: 10.18653/v1/2020.acl-main.708. URL `https://aclanthology.org/2020.acl-main.708`.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2096–2111, 2020b. doi: 10.18653/v1/2020.findings-emnlp.190. URL https://aclanthology.org/2020.findings-emnlp.190/.

Zhoujun Cheng, Wenxuan Zhou, Yue Dong, Yuning Mao, Zhilin Yan, Yifan Ethan Zhang, Mu Li, Muhao Chen, and Xiang Ren. De-confounded variational encoder-decoder for logical table-to-text generation. 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)*, pages 5847–5859, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.455. URL https://aclanthology.org/2021.acl-long.455/.

Zhoujun Cheng, Wenxuan Zhou, Yue Dong, Yuning Mao, Zhilin Yan, Chen Zhao, Yifan Ethan Zhang, Xinyi Zheng, Mu Li, Muhao Chen, and Xiang Ren. Hitab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1204, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.85. URL https://aclanthology.org/2022.acl-long.85/.

Junyoung Chung, Çağlar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. doi: 10.48550/arXiv.1412.3555. ICLR 2015 Workshop Track.

Shumin Deng, Jiacheng Yang, Hongbin Ye, Chuanqi Tan, Mosha Chen, Songfang Huang, Fei Huang, Huajun Chen, and Ningyu Zhang. Logen: few-shot logical knowledge-conditioned text generation with self-training. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2124–2133, 2023.

Heng Gong, Wei Bi, Xiaocheng Feng, Bing Qin, Xiaojiang Liu, and Ting Liu. Enhancing content planning for table-to-text generation with data understanding and verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2920–2930, 2020. doi: 10.18653/v1/2020.findings-emnlp.262. URL https://aclanthology.org/2020.findings-emnlp.262/.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, 2016. doi: 10.18653/v1/P16-1154. URL https://aclanthology.org/P16-1154.

Lei He, Nabil Aouf, and Bifeng Song. Explainable deep reinforcement learning for UAV autonomous path planning. *Aerospace Science and Technology*, 118: 107052, October 2021. doi: 10.1016/j.ast.2021.107052. URL https://doi.org/10.1016/j.ast.2021.107052.

Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214:106685, January 2021. doi: 10.1016/j.knosys.2020.106685. URL https://doi.org/10.1016/j.knosys.2020.106685.

Thomas Hickling, Abdelhafid Zenati, Nabil Aouf, and Phillippa Spencer. Explainability in deep reinforcement learning: A review into current methods and applications. *ACM Computing Surveys*, 56(5):1–35, September 2023. doi: 10.1145/3603229. URL https://doi.org/10.1145/3603229.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Victoria J. Hodge, Richard Hawkins, and Rob Alexander. Deep reinforcement learning for drone navigation using sensor data. *Neural Computing and Applications*, 33(6):2015–2033, March 2021. doi: 10.1007/s00521-020-05081-8. URL https://doi.org/10.1007/s00521-020-05081-8.

Zijian Hu, Xiaoguang Gao, Kaifang Wan, Yiwei Zhai, and Qianglong Wang. Relevant experience learning: A deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments. *Chinese Journal of Aeronautics*, 34(12):187–204, December 2021. doi: 10.1016/j.cja.2021.06.012. URL https://doi.org/10.1016/j.cja.2021.06.012.

Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Explainable Artificial Intelligence*, pages 47–53, 2019. URL https://web.engr.oregonstate.edu/~afern/papers/reward_decomposition__workshop_final.pdf.

Katherine A. Keith, David Jensen, and Brendan O'Connor. Text and causal inference: A review of using text to remove confounding from causal estimates. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5332–5344, 2020. doi: 10.18653/v1/2020.acl-main.474. URL https://aclanthology.org/2020.acl-main.474.

Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. In *Proceedings of the*

2016 Conference on Empirical Methods in Natural Language Processing, pages 1203–1213, 2016. doi: 10.18653/v1/D16-1128. URL https://aclanthology.org/D16-1128.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Liang Li, Can Ma, Yinliang Yue, and Dayong Hu. Improving encoder by auxiliary supervision tasks for table-to-text generation. 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)*, pages 5979–5989, 2021. doi: 10.18653/v1/2021.acl-long.466. URL https://aclanthology.org/2021.acl-long.466/.

Percy Liang, Michael I. Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, 2009. doi: 10.3115/1687878.1687893. URL https://aclanthology.org/P09-1011.

Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. Plog: Table-to-logic pretraining for logical table-to-text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*, pages 5517–5530, 2022. doi: 10.18653/v1/2022.emnlp-main.373. URL https://aclanthology.org/2022.emnlp-main.373/.

Baoqiang Liu, Yu Bai, Fang Cai, Shuang Xue, Na Ye, and Xinyuan Ye. Reasoning knowledge transfer for logical table-to-text generation. In *Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Yokohama, Japan, jul 2024. IEEE. doi: 10.1109/IJCNN60899.2024.10649960.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 4881–4888. AAAI Press, 2018. doi: 10.1609/aaai.v32i1.11925. URL https://doi.org/10.1609/aaai.v32i1.11925.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the*

2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, 2015. doi: 10.18653/v1/D15-1166. URL https://aclanthology.org/D15-1166.

Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 2493–2500, 2020. doi: 10.1609/aaai.v34i03.5631. URL https://ojs.aaai.org/index.php/AAAI/article/view/5631.

Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. SciGen: A dataset for reasoning-aware text generation from scientific tables. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Track on Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/149e9677a5989fd342ae44213df68868-Paper-round2.pdf. Datasets and Benchmarks Track (Round 2).

Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889, 2018. doi: 10.18653/v1/D18-1422. URL https://aclanthology.org/D18-1422.

OpenAI. OpenAI chat completions API. https://platform.openai.com/docs/guides/text-generation/chat-completions-api, 2020.

Suixin Ou and Yongmei Liu. Learning to generate programs for table fact verification via structure-aware semantic parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7624–7638, 2022. doi: 10.18653/v1/2022.acl-long.525. URL https://aclanthology.org/2022.acl-long.525.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Judea Pearl. On measurement bias in causal inference. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI 2010)*, pages 425–432, Catalina Island, California, USA, 2010. AUAI Press. URL https://ftp.cs.ucla.edu/pub/stat_ser/r357.pdf.

Yotam Perlitz, Liat Ein-Dor, Dafna Sheinwald, Noam Slonim, and Michal Shmueli-Scheuer. Diversity enhanced table-to-text generation via type control. *CoRR*, abs/2205.10938, 2022. doi: 10.48550/arXiv.2205.10938.

Raheel Qader, Khoder Jneid, François Portet, and Cyril Labbé. Generation of company descriptions using concept-to-text and text-to-text deep models: Dataset collection and systems evaluation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 254–263, 2018. doi: 10.18653/v1/W18-6534. URL https://aclanthology.org/W18-6534.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Technical Report*, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Swarnadeep Saha, Xinyan Velocity Yu, Mohit Bansal, Ramakanth Pasunuru, and Asli Celikyilmaz. Murmur: Modular multi-step reasoning for semi-structured data-to-text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11072–11087, 2023. doi: 10.18653/v1/2023.findings-acl.704. URL https://aclanthology.org/2023.findings-acl.704/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, July 2017. URL https://arxiv.org/abs/1707.06347.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, 2017. doi: 10.18653/v1/P17-1099. URL https://aclanthology.org/P17-1099.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. Towards table-to-text generation with numerical reasoning. 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)*, pages 1451–1465, 2021. doi: 10.18653/v1/2021.acl-long.115. URL https://aclanthology.org/2021.acl-long.115.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2nd edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Deepak-George Thomas, Daniil Olshanskyi, Karter Krueger, Tichakorn Wongpiromsarn, and Ali Jannesari. Interpretable uav collision avoidance using deep reinforcement learning. In *CoRR abs/2105.12254*, page –, 2021. URL https://arxiv.org/abs/2105.12254.

Guo Tong, Nan Jiang, Biyue Li, Xi Zhu, Ya Wang, and Wenbo Du. UAV navigation in high dynamic environments: A deep reinforcement learning approach. *Chinese Journal of Aeronautics*, 34(2):479–489, February 2021. doi: 10.1016/j.cja.2020.08.029. URL https://doi.org/10.1016/j.cja.2020.08.029.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:6000–6010, 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

George A. Vouros. Explainable deep reinforcement learning: State of the art and challenges. *ACM Computing Surveys*, 55(5):1–39, October 2022. doi: 10.1145/3527448. URL https://doi.org/10.1145/3527448.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pages 24824–24837. Curran Associates, Inc., December 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/5f1461563c3b2abc5a0d44d545d5fba0-Paper-Conference.pdf.

Chao Yan, Xiaojia Xiang, and Chang Wang. Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments. *Journal of Intelligent & Robotic Systems*, 98:297–309, January 2020. doi: 10.1007/s10846-019-01093-z. URL https://doi.org/10.1007/s10846-019-01093-z.

Bohao Yang, Chen Tang, Kun Zhao, Chenghao Xiao, and Chenghua Lin. Effective distillation of table-based reasoning ability from LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5538–5550, 2024. doi: 10.18653/v1/2024.lrec-main.492. URL https://aclanthology.org/2024.lrec-main.492.

Ori Yoran, Alon Talmor, and Jonathan Berant. Turning tables: Generating examples from semi-structured tables for endowing language models with reasoning skills. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6016–6031, 2022. doi: 10.18653/v1/2022.acl-long.416. URL https://aclanthology.org/2022.acl-long.416.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. doi: 10.48550/arXiv.2205.01068.

Xueliang Zhao, Tingchen Fu, Lemao Liu, Lingpeng Kong, Shuming Shi, and Rui Yan. Sortie: Dependency-aware symbolic reasoning for logical data-to-text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11247–11266, 2023a. doi: 10.18653/v1/2023.findings-acl.715. URL https://aclanthology.org/2023.findings-acl.715/.

Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. ReasTAP: Injecting table reasoning skills during pre-training via synthetic reasoning examples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9006–9018, 2022. doi: 10.18653/v1/2022.emnlp-main.615. URL https://aclanthology.org/2022.emnlp-main.615/.

Yilun Zhao, Zhenting Qi, Linyong Nan, Lorenzo Jaime Yu Flores, and Dragomir Radev. Loft: Enhancing faithfulness and diversity for table-to-text generation via logic form control. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023)*, pages 554–561, 2023b. doi: 10.18653/v1/2023.eacl-main.40. URL https://aclanthology.org/2023.eacl-main.40/.

Yilun Zhao, Zhenting Qi, Linyong Nan, Boyu Mi, Yixin Liu, Weijin Zou, Simeng Han, Ruizhe Chen, Xiangru Tang, Yumo Xu, Dragomir Radev, and Arman Cohan. QTSumm: Query-focused summarization over tabular data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1157–1172, 2023c. doi: 10.18653/v1/2023.emnlp-main.74. URL https://aclanthology.org/2023.emnlp-main.74.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios. In *Proceedings of*

the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 160–175, Singapore, December 2023d. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-industry.17. URL https://aclanthology.org/2023.emnlp-industry.17/.

Ender Çetin, Cristina Barrado, and Enric Pastor. Explainability of deep reinforcement learning method with drones. In *Proceedings of the IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, page —, 2023. doi: 10.1109/DASC58513.2023.10311156. URL https://ieeexplore.ieee.org/document/10311156.

Appendix A

Appendix

A.1 Logic Type

The following table shows the logic types and their functions, arguments, and output.

Logic Type	Name	Arguments	Output
Selection	filter_eq/not_eq	view, header string, value	view
	filter_greater/less	view, header string, number	view
	hop	row (or view,row-id), header string	object
Multi-dim / hierarchical selection	region_by_path	table, header path	region/view
	filter_tree	region, tree predicate	region
Aggregation	sum/avg	view, header string	number
Arithmatic	diff	number, number	number
	div	number, number	number
	pct_change	new number, old number	number
Counting	count	view	number
Only	only	view, predicate	bool
Majority	majority	view, predicate	bool
Comparison	eq/not_eq	object, object	bool
	greater/less	number, number	bool
	round_eq	number, number, tol	bool
Superlative	argmax/argmin_row	view, header string	view (row)
	max_value/min_value	view, header string	number
Ordinal	kth_argmax/kth_argmin	view, header string, int k	view (row)
	rank_of_row	view, header string, row	int
Unique (distinct)	unique_values	view, header string	list of values
	count_distinct	view, header string	number
Temporal	date_diff	date, date	duration/number
	earliest/latest	list of dates	date
	trend	list of dates, list of numbers	label
Structural/ hierarchical	group_by_header	table, header level	list of views/regions
	virtual_entity	list of views/regions	view/region
Logical composition	and	bool, bool	bool
	or	bool, bool	bool
	not	bool	bool

Table A.1: LT2T Logic Types

A.2 Taxonomy of error type

These tables provide definitions and examples of error types that occur in each logic type.

Aggregation		
Error type	**Definition**	**Example**
Average mis-calculation	Mean computed incorrectly.	Scores $[8, 10, 14] \rightarrow 10.67$, model returns 12.
Sum mis-calculation	Incorrect addition of values.	Sales $[120, 250, 130] \rightarrow 500$, model outputs 490.
Rounding/tolerance	Correct value before rounding, but the comparison uses the wrong tolerance.	True 21.57; model rounds to 21.4 and treats as equal.
Row-filtering issue	Wrong row selection.	Query "avg salary *in 2024*", model averages *all years*.
Comparative		
Numeric-diff mismatch	Difference is right, but string/precision format is off.	Correct diff 4, model emits "4.0"; strict check fails.
Unsupported op./format	Executor lacks needed op. (date arithmetic, mixed units).	Needs "2024-05-01 – 2023-05-01", executor returns error.
Unit-token mismatch	Same number, different unit token.	Gold "6 *points*", model outputs "6".
Composite-logic failure	Faulty sub-clause in `AND`/`OR` flips the result.	"(diff>5 **AND** team='A')" diff ok, team check fails.

Table A.2: Failure patterns for Aggregation and Comparative logic types.

Count		
Error type	**Definition**	**Example**
Under-count	Some qualifying rows missed.	12 rows with *Status=Completed*; model counts 10.
Over-count	Extra rows included.	Counts "Completed" *and* "Done" $\rightarrow$ 14.
Substring filter issue	Substring match too broad/narrow.	"title CONTAINS 'Cup'" also matches "Coupe".
Parser/format mismatch	Number/date format mismatch break equality.	Gold "15 July"; model parses "July 15".
Majority		
Fuzzy vs. strict equality	Case/whitespace diff triggers false mismatch.	Majority city "new york" vs "New York".
Off-by-one threshold	Uses wrong $>$ 50% criterion.	3/5 rows $\rightarrow$ majority True; model requires ≥ 4.
Comparator-/argument-swap	Predicate arguments reversed.	Tests "away = home?" instead of "home = away?".

Table A.3: Failure patterns for Count and Majority logic types.

Ordinal		
Error type	**Definition**	**Example**
Wrong ordinal index/tie	Chooses wrong *nth* or mishandles ties.	Needs 2nd-lowest price (tie), model picks first seen entry.
Parser/format mismatch	Mis-parsed number string.	Rank "02" vs "2".
Ordinal, Superlative & Only errors		
ONLY filter size error	`ONLY` should leave exactly 1 row; returns 0 or >1.	"only product > $1000" → filter returns three products.
Row-filtering pre-filter loose	MAX/MIN computed over unfiltered set.	"max speed for Team X", model uses all teams.
Fuzzy vs. strict equality	String/format mismatch inside ordinal/superlative.	Expected "3rd", model outputs "3"; or "M. Rose" vs "Mauri Rose".

Table A.4: Failure patterns for Ordinal, Superlative, and Only logic types.