From Tables to Quantified Statements: Evaluating LLM Inference Generation through Executable Verification

Mai Mohamed Eida, Gunjan Anand, Ayush Singh, Aleksandre Maskharashvili
University of Illinois Urbana Champaign
{maimm2, gunjana2, ayushs13, am172}@illinois.edu

Abstract

LLMs can generate fluent descriptions from tables, but their outputs may remain logically unsupported by the structured data. We introduce STAT-TO-TEXT[1], a controlled task in which LLMs generate quantified natural language inferences from statistical tables using quantified constructions such as *all*, *some*, *no*, and *most*. To evaluate these inferences, we use an LLM generated Python checker code which when executed verifies the corresponding truth conditions against the table. We compare four open-weight LLMs across model families and scales, evaluating faithfulness, logical accuracy, table coverage, and diversity. Our results show that model scale and family matter, with the largest model (GPT-OSS-120B) consistently producing the most faithful inferences without sacrificing greater table coverage and quantifier diversity, as opposed to smaller models. These findings are supported by human annotation, which shows that the automated checker closely aligns with human judgments.

1 Introduction

Data-to-text generation aims to render structured inputs, such as tables (Wang et al., 2021), graphs (Song et al., 2020), records, and meaning representations, as accurate and faithful natural language (Reiter and Dale, 2000; Lin et al., 2024). Traditional Natural Language Generation (NLG) pipelines decompose generation into content selection, sentence planning, and surface realization, whereas recent neural language models integrate these stages through end-to-end (Yin and Wan, 2022), fine-tuning (Gong et al., 2019), or prompt-based generation.

In this work, we focus on table-to-text generation, where the input is a table containing numerical and categorical variables. Large language models (LLMs) now achieve strong performance on structured data-to-text tasks without task specific fine-tuning or large collections of human written references (Zhao et al., 2023). In-context learning and chain-of-thought prompting can further support structured inputs through reasoning (Wei et al., 2022; Zhao et al., 2023). Nevertheless, fluent outputs may be unsupported by the source table, overlook relevant rows or variables, overgeneralize from a small subsets of records, misrepresent numerical relations, or repeat near-identical claims. Faithfulness, hallucinations, factual accuracy, coverage, and diversity therefore remain central challenges in LLM based table-to-text generation (Calò et al., 2026; Kasner et al., 2021).

Although prior table-to-text research has incorporated numerical and logical operations (Liu et al., 2021; Zhao et al., 2023; Trigg and Hougen, 2025) such as aggregation, comparison, superlatives, and majority reasoning (Suadaa et al., 2021; Moosavi et al., 2021; Chen et al., 2020a,b; Trigg and Hougen, 2025), generalized quantifier selection has received comparatively limited attention as a primary generation and evaluation objective. A generalized quantifier, such as *all*, *some*, *no*, *most*, *at least half*, *more than two-thirds*, *exactly three*, and *all but one* (Barwise and Cooper, 1981), expresses a relation between two sets: the *restrictor* (R), which defines the domain of quantification, and the *nuclear scope* (henceforth *scope* – S), which specifies the property attributed to that domain. Generalized quantifiers are particularly well suited to statistical table-to-text generation as they can represent patterns spread across the table as brief, understandable natural language statements. This enables users to grasp overall statistical observations without examining each table row individually. At the same time, they impose precise semantic constraints that can be formally verified: for example, in *Most people over 70 are self-employed*

[1] STAT-TO-TEXT Full code and results are released on GitHub.

Proceedings of the 19th International Natural Language Generation Conference: Long Papers, pages 351–368
October 17-21, 2026 ©2026 Association for Computational Linguistics

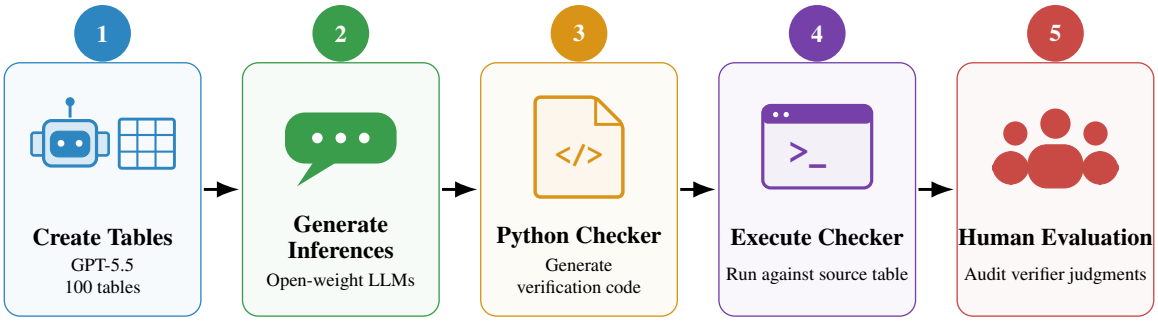


Figure 1: STAT-TO-TEXT: Quantified inference generation and evaluation pipeline. An LLM generates both quantified inferences and Python checker scripts based on statistical tables and inferences respectively. Python checkers scripts are executed on the source tables, and a sampled subset is evaluated by human annotators to validate the LLM inference verification output.

or retired the standard interpretation of *most* requires more than half of R (*people over 70*) be members of S (*self-employed or retired*).

Generating such statements therefore calls for filtering, counting, comparison, aggregation, and quantifier selection. To the best of our knowledge, prior work has not systematically examined whether LLM generated quantified inferences are supported by a fixed statistical table or whether models distinguish the truth conditions associated with generalized quantifiers.

In this paper we focus on a form of table-to-text generation, which we call STAT-TO-TEXT: generating faithful quantified natural language statements inferred from a *statistical table*, i.e., table of numerical and categorical variables from which meaningful statistics can be drawn. The target output is not a general summary or a verbalization of selected cells, but a set of quantified statements over table defined row sets. For example, given a table containing "age" and "employment" status, a model may generate *Most people over 70 are self-employed or retired*. After generating inferences, our pipeline, illustrated in Figure 1, uses an LLM generated Python checker to verify the faithfulness of the inferred statements. Then we evaluate the quality of the results quantitatively by checking faithfulness, coverage, and diversity of the generated statements, followed by a qualitative evaluation through human annotation.

In this work, we address the following research questions:

RQ1. *How reliably does our* STAT-TO-TEXT *pipeline generate and verify quantified natural language inferences from statistical tables?*

RQ2. *How do model family, model scale, and reasoning paradigm (direct prediction vs. reasoning models) affect the logical faithfulness of quantified* STAT-TO-TEXT *generation?*

RQ3. *How do LLMs differ in the coverage and diversity of the quantified inferences they generate?*

Our results show that the STAT-TO-TEXT pipeline reliably generates and verifies faithful quantified inferences from statistical tables, with the limitation of not yet achieving 100% throughput, as some tables are dropped during inference generation or checker execution. We further find that both model family and model scale influence logical faithfulness, with GPT-OSS-120B achieving the highest faithfulness without sacrificing overall table and quantifier coverage, and with reasonable variable and quantifier diversity. This is strengthened by GPT-OSS-120B maintaining a near perfect agreement with human annotators (97.5%), the highest of all models.

2 Related Work

2.1 Table-to-Text Generation and Table Reasoning

Table-to-text generation has evolved from modular NLG pipelines to neural and LLM-based systems that generate directly from tabular inputs (Lin et al., 2024; Liu et al., 2022). Early neural benchmarks emphasized fluent generation and content selection (Castro Ferreira et al., 2019), while later work increasingly focused on factual grounding and reasoning over multiple table records, such as RotoWire (Wiseman et al., 2017) and ToTTo (Parikh et al., 2020). Both benchmarks focus on generating natural language descriptions from table content rather than generating analytical or quantified inferences.

Recent work extends table-to-text generation from cell verbalization toward numerical infer-

ence (Onderková et al., 2025). Both NumericNLG (Suadaa et al., 2021) and SciGen (Moosavi et al., 2021) focus on descriptions of numerical reasoning over values in tables from scientific papers, where such descriptions require arithmetic and comparative reasoning. In contrast, approaches such as Pérez et al. (2025) generate textual insights from multi-table databases through an intermediate SQL-based reasoning step, rather than by directly prompting an LLM over the table.

Logical table-to-text generation complements this numerical perspective. LogicNLG (Chen et al., 2020a) and Logic2Text (Chen et al., 2020b) formulate generation as producing natural language statements that are logically entailed by open-domain semistructured tables. They pair table grounded descriptions with logical forms and cover operations such as comparison, aggregation, superlatives, majority, and uniqueness. These tasks are closely related to ours, but they treat such relations as part of a broader inventory of logical operations. We instead isolate generalized quantifier selection and evaluate whether the generated quantifier correctly expresses the relation over the table.

LLMs have further broadened table-to-text generation beyond fixed benchmark settings. Zhao et al. (2023) evaluate LLMs as generators, evaluators, and feedback providers for table insight and query-based generation tasks, while Trigg and Hougen (2025) survey logical table-to-text generation. This work shows that prompted LLMs can perform flexible table interpretation without task specific training, and identify reasoning, numeracy, faithfulness, and verification as central challenges. While these studies motivate the use of LLMs for table based inferences, it remains open how reliably models generate quantified claims whose truth depends on relations distributed across multiple rows.

2.2 Generalized Quantifiers

Generalized quantifiers are analyzed in formal semantics as relations between sets (Barwise and Cooper, 1981). For our running example, *Most people over 70 are self-employed or retired*, the standard interpretation of *most* requires more than half of R belong to S:

$$\text{most}(R, S) \iff |R \cap S| > \tfrac{1}{2}|R|$$

This interpretation maps naturally onto tabular data. Rows define entities; headers serve as attributes used to construct predicates on numerical and categorical variables, giving rise to restrictors and scopes. Proportional quantifiers such as *most* and *at least half* depend on the ratio of $|R \cap S|$ to $|R|$; universal quantifiers such as *all* require $R \subseteq S$; existential quantifiers such as *some* require $R \cap S \neq \emptyset$; negative quantifiers such as *no* require $R \cap S = \emptyset$; and cardinal quantifiers such as *exactly three* depend on the size of $R \cap S$. Generating a faithful quantified statement therefore means identifying the restrictor and scope predicates, computing the corresponding set relation, and selecting a linguistic expression with matching truth conditions.

When it comes to quantified statements in NLG, Chen et al. (2019) study quantified descriptions of abstract visual scenes and evaluate whether generated descriptions are correct, complete, and human-like. We transfer this problem to tabular data, where the restrictor and scope must be constructed from numerical and categorical table attributes and the selected quantifier must be licensed by the corresponding row counts.

2.3 LLM Python Verification

Faithfulness remains a central limitation of table-to-text generation. This gap has been addressed through entity centric perspectives (Liu et al., 2021), token level detection of incorrect values and statements (Kasner et al., 2021), and logical consequence of unsupported outputs (Calò et al., 2026).

However, another method to address faithfulness is through Program-aided language models (PAL) (Gao et al., 2023), where it delegates symbolic and numerical operations to executable code. In this paper, we use PAL to rely on execution-based reasoning, and we adapt this idea to evaluation, translating each generated inference into a Python verification function that is executed against the source table to address faithfulness. The reliability of other automatic table-to-text evaluation is an open problem: automatic metrics do not always agree with LLM-based or human judgments of table generated insights (Onderková et al., 2025). This motivates evaluating the semantic content using PAL of individual statements rather than relying on exclusively learned metrics which are not suitable for STAT-TO-TEXT evaluation. Our work therefore connects numerical table-to-text generation, logical table reasoning, quantified expression generation, and factuality evaluation.

3 Data Setup

Since our experiments require previously unseen tables with controlled row and header counts, we prompt GPT-5.5 to generate a synthetic dataset of tabular data. We select a different model for table generation than those used in the next stages of our pipeline to ensure consistency across all synthetic tables while avoiding potential bias toward any of the models evaluated in our experiments. We generate 100 tables, each containing 5 - 10 variables and 15 rows, using the prompt illustrated in Appendix A. The topics of these tables range across the following themes: healthcare, retail/business, transportation/logistics, household finance, sports, agriculture, and HR/employee management, as illustrated in Table 1. A full sample table is shown in Table 4 in Appendix B.

Fixing rows and number of variables allows us to control for table complexity, ensuring the evaluation results can be attributed to the models rather than variations in the source table size.

4 STAT-TO-TEXT Pipeline Setup

Step 1 – LLM Inferences: After generating our statistical tables, we prompt all selected LLMs to generate quantified inferences for each statistical table. We limit the inferences to 20 inferences, and specify a diverse set of universal, existential, majority, negation, and conditional statements in the few-shot prompting. The same prompt is used across both direct prediction models and reasoning models (Figure A.2 in Appendix A). Therefore, models generate inferences similar to the following (more examples in Appendix C):

1. All individuals in the table have a systolic blood pressure between 112 and 156 mmHg.
2. For all individuals with a BMI greater than 30, their age is greater than 40 years.

Specifying a fixed number of generated inferences is crucial at this stage to preserve inference quality, otherwise, some models generate 100+ inferences until the token limit is reached, despite explicit instruction to only generate 20 meaningful inferences. However, some reasoning models skip inference generation after becoming stuck in the reasoning process and exhausting the available tokens, dropping tables in the process. After generating 20 inferences for all the tables in the dataset, we move to step 2 in the pipeline: an LLM generated Python Checker.

Step 2 – LLM Generated Python Checker: We prompt the model to generate a Python script that automatically verifies the factual correctness of each generated inference with respect to its corresponding table. In the prompt, we provide the table header (variables) and the list of inferences generated in step 1 for each table. We omit the table cell values to prevent contaminating the script generation step from relying on specific values rather than producing a generalizable verification script.

The script contains one verification function per inference, where each function evaluates whether the statement is true or false by querying the table using the pandas library. In addition to the binary verdict, the script outputs a brief justification for its decision, as illustrated in Listing 1. This automated verification allows us to assess the factual consistency of the generated inferences, while ensuring that every inference is evaluated using the same procedure.

```python
def stmt_1(df: pd.DataFrame):
    """All individuals have systolic blood pressure
    between 112 and 156 mmHg."""

    local = df.copy()
    local["bp_systolic"] = pd.to_numeric(
        local["bp_systolic"], errors="coerce"
    )

    mask = local["bp_systolic"].between(112, 156)
    truth = bool(mask.all())

    if truth:
        return True, "No violations found."
    else:
        return False, (
            f"Found {len(local[~mask])} violating rows."
        )
```

Listing 1: Excerpt of an automatically generated Python verification function.

In this stage, we developed two prompt variants to account for differences in prompt sensitivity between direct-prediction models and reasoning models. Reasoning models generally produced valid verification scripts from concise prompts (Figure A.4 in Appendix A). However, direct-prediction models often required additional implementation constraints to improve consistency and reduce common code generation errors across models (Figure A.3 in Appendix A).

5 Model Selection

We evaluate four open-weight LLMs spanning two model families and two parameter scales, including both direct-prediction and reasoning-oriented models: LLAMA 3.1 8B, LLAMA 3.1 70B (Grattafiori et al., 2024), GPT-OSS 20B, and GPT-OSS 120B (OpenAI, 2025). These models

Patient ID	Diagnosis	Age	Systolic BP	Diastolic BP	Chol.	BMI	Smoker
PT001001	asthma	46	129	72	192	33.3	no
PT001002	diabetes	62	143	82	223	22.5	yes
PT001003	diabetes	68	147	87	198	31.0	no
PT001004	arthritis	46	135	77	182	27.6	yes
PT001005	hypertension	76	119	74	245	22.7	no

Table 1: Example of a synthetic statistical table generated by GPT-5.5.

were selected to examine the effects of both model family and model scale on inference generation and verification performance. We focus on open-weight models to enable reproducibility and to limit computation cost associated with repeatedly querying proprietary APIs throughout the evaluation pipeline. The only exception is GPT-5.5 (OpenAI, 2026), which is used exclusively for synthetic statistical table generation, as justified in Section 3.

6 Automated Evaluation

At this stage, we have executed the pipeline and need to evaluate the quality of the generated inferences. We adapt previous table-to-text evaluation metrics (Liu et al., 2021; Lin et al., 2024) and introduce additional metrics tailored to quantified inference generation. For a set of inferences to be informative, they must first be faithful: grounded in the source table without referencing external information, and logically accurate with respect to the table. They must also exhibit high coverage by representing both the table headers and rows, while covering full generalized quantifier categories (e.g., universal, existential, conditional, and negation). Finally, the generated inferences should be diverse. We therefore measure both variable redundancy and quantifier redundancy to ensure that the model does not repeatedly focus on the same variables or lexical realizations of quantifiers. Our results are illustrated in Table 2.

6.1 Faithfulness

Grounding is measured through whether the inferences refer only to variables, entities, categories, and values present in the source table. **Accuracy** measures whether a generated statement is true with respect to the source table: a statement is accurate when the relation expressed by its predicates and quantifier satisfies the corresponding truth conditions over the relevant table cells. Taken together, these metrics provide a measurement for faithfulness (Liu et al., 2021).

Results All models achieve near-perfect grounding (99.5–100%), which means tables are not generating statements which references external values. This is a desirable outcome, however, taken with the accuracy results (LLAMA-8B accuracy rate is 48.7% despite achieving 100% in grounding), it seems that grounding alone is insufficient to ensure logically valid (accurate) quantified inferences. Scale does matter for faithfulness, with GPT-OSS-120B achieving the best faithfulness scores.

6.2 Coverage

Table Record Coverage (TRC) quantifies the extent to which the generated inferences utilize the information contained in the rows of the input table. TRC is computed as the number of unique table rows referenced by at least one inference divided by the total number of rows in the table. A value of 1 indicates that every table row contributes to at least one generated inference, whereas lower values indicate that only a subset of the table is represented in the generated inferences.

$$\mathrm{TRC} = \frac{\left|\bigcup_{i=1}^{N} R_i\right|}{|T|}$$

where T denotes the set of all table rows, R_i is the subset of rows referenced by the i-th inference, and N is the total number of generated inferences.

Results As illustrated in Table 2, our results show that all models achieve high TRC, ranging from 92–100% across the generated inferences. GPT-OSS-120B and LLAMA-70B achieve the highest coverage, reaching 100% and 99%, respectively. Model scale impacts row coverage, though the difference in impact is negligible.

Variable Coverage (VC) measures the extent to which the generated inferences utilize the variables present in the input table. A variable is captured if it is referenced by name or by a column value in at least one inference. If the inference mentions the column value "diabetes" it would consider the

variable "diagnosis" as being captured.

$$\mathrm{VC} = \frac{\left|\bigcup_{i=1}^{N} V_i\right|}{|V|}$$

where V is the set of all variables (columns) in the input table, V_i is the subset of variables referenced by the i-th inference, and N is the total number of generated inferences. VC, taken together with TRC, illustrates the entire coverage of the table rows and columns. Higher values indicate that the generated inferences make use of a broader range of the available information.

Results VC scores vary between 87–95% (Table 2), showing lower coverage of columns than for TRC scores for rows. However, VC closely mirrors the trends observed for TRC in relation to scale, indicating that larger models maintaining broader row coverage also preserve more comprehensive coverage of the underlying table variables with LLAMA-70B (94.8%) and GPT-OSS-120B (95.6%) achieving the highest VC coverage.

Quantifier Coverage (QC) evaluates the use of all generalized quantifiers categories across the generated inferences. We analyze this from two perspectives. First, we evaluate Quantifier Coverage (QC), which measures whether the model uses the full categories of generalized quantifiers (Table 2). Specifically, we calculate the number of distinct quantifier categories used in the 20 inferences divided by the five predefined categories (universal, existential, majority/frequency, negation, and conditional). Second, we examine the distribution of quantifier categories to determine whether models favor certain quantifiers over others. To do so, we categorize each generated inference according to its primary quantifier category and report the frequency of each category. The resulting distributions are illustrated in Figure 2.

Results QC, (Table 2), is high across all models but is consistently highest for the LLAMA family in both coverage and distribution. This shows that the full set of generalized quantifier categories prompted is consistently represented across the generated inferences, despite the prompt providing only examples rather than explicitly requiring each quantifier category. However, higher QC does not correspond to greater faithfulness. Although the LLAMA models consistently use a wider range of quantifier categories (LLAMA-70B, QC = 99.1%), the GPT-OSS models produce substantially more faithful quantified inferences (Grounding $\geq$ 95% and Accuracy $\geq$ 90%). Further, GPT-OSS-20B achieves high accuracy despite its smaller scale; however, this appears to come with a trade-off, as its quantifier distribution is more heavily skewed toward universal quantifiers (Figure 2). This suggests that generating a broader variety of quantifier categories is largely independent of accurately applying their corresponding truth conditions.

6.3 Diversity

Variable Redundancy (VR) measures the extent to which generated inferences repeatedly reference the same table variables. While some repetition is expected, excessive reuse indicates that the model concentrates on a smaller subset of variables instead of generating inferences that make use of the full table. For each table t, let V_t denote the set of unique variables (columns), and let $f_t(v)$ denote the number of generated inferences that reference variable v. We define variable redundancy as

$$\mathrm{VR}_t = \frac{\sum_{v \in V_t} \max\left(f_t(v) - |V_t|,\, 0\right)}{\sum_{v \in V_t} f_t(v)}.$$

The first $|V_t|$ occurrences of each variable are not considered redundant, while each additional occurrence contributes to the numerator. We use $|V_t|$ as the redundancy threshold because some variable reuse is unavoidable. For example, a table with six variables and twenty generated inferences cannot describe every inference using entirely distinct variables, making repeated references inevitable. By allowing each variable to appear up to $|V_t|$ times before counting additional occurrences as redundant, the metric distinguishes expected reuse from excessive concentration on a small subset of variables.

A value of 0 indicates that no variable is referenced more than $|V_t|$ times within the generated inferences for a table. Higher values indicate greater repetition of the same variables and a stronger concentration of the generated inferences on a smaller subset of the available table variables. We calculate VR_t separately for each table and report the mean across all tables for each model.

Results VR remains low across all models (Table 2), indicating that the generated inferences are generally distributed across the available table variables rather than concentrating on only a few. GPT-OSS-20B achieves the lowest redundancy (13.2%), while LLAMA-70B exhibits the highest redundancy (16.8%). Unlike Variable Coverage, which

	LLM Inferences AFTER Python Checker							
	# Tables	**Grounding↑**	**Accuracy↑**	**TRC↑**	**VC↑**	**VR↓**	**QC↑**	**QR↓**
LLAMA-8B	86	**1**	0.487	0.95	0.878	0.149	0.9	**0.805**
LLAMA-70B	94	0.999	0.713	0.994	0.948	0.168	**0.991**	0.948
GPT-OSS-20B	78	0.96	0.873	0.929	0.909	**0.132**	0.754	0.89
GPT-OSS-120B	95	0.995	**0.938**	**1**	**0.956**	0.159	0.956	0.822

Table 2: Performance of open-weight LLMs on STAT-TO-TEXT generated inferences' faithfulness (grounding and accuracy), coverage (TRC, VC, QC), and diversity (VR, QR) *after* Python-based verification. Metrics annotated with ↑ indicate that higher values correspond to better performance, whereas metrics annotated with ↓ indicate that lower values are preferred. STAT-TO-TEXT does not achieve 100% throughput, as some tables are dropped during Steps 1 or 2 of the pipeline, resulting in the number of remaining tables shown in the Table column.

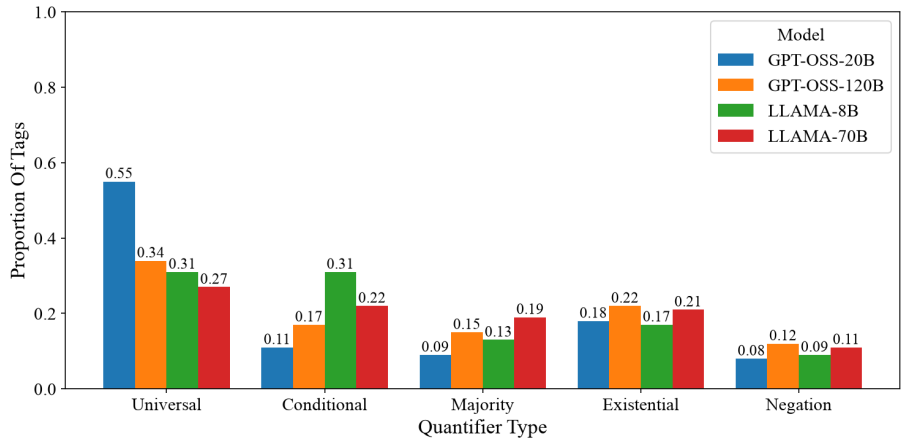


Figure 2: Distribution of quantifier categories generated by each model across the synthetic statistical tables. Bars represent the proportion of generated inferences assigned to each quantifier category (Universal, Conditional, Majority, Existential, and Negation), aggregated over all generated outputs. Values above each bar indicate the corresponding proportion.

measures the breadth of variables represented, VR captures how evenly the generated inferences are distributed over those variables. Together, these metrics suggest a trade-off: larger models achieve broader variable coverage while exhibiting slightly higher redundancy, whereas smaller models reference variables more evenly but cover a narrower subset of the available variables.

Quantifier Redundancy (QR) measures lexical concentration within each quantifier category. We use for it normalized Herfindahl–Hirschman Index. Let L_t denote the set of available lexical realizations for quantifier category t, such as *all*, *every*, *each*, and *any* for universal quantification. If $f_{t,\ell}$ is the frequency of lexical realization ℓ and $N_t = \sum_{\ell \in L_t} f_{t,\ell}$, quantifier redundancy for category t is defined as

$$\mathrm{QR}_t = \frac{\sum_{\ell \in L_t} (f_{t,\ell}/N_t)^2 - 1/|L_t|}{1 - 1/|L_t|}.$$

The metric ranges from 0 to 1. A score of 0 indicates that the available lexical realizations ("all", "every") are distributed evenly within each quantifier category (universal quantifiers), reflecting high lexical variety, whereas a score of 1 indicates that a single realization accounts for all occurrences of that quantifier category. The overall quantifier redundancy score for a table is calculated as the frequency-weighted average across quantifier categories.

Results As illustrated in Table 2, Quantifier Redundancy (QR) varies considerably across models. LLAMA-8B achieves the lowest QR (80.5%), indicating the greatest lexical variety in expressing quantifiers but this comes at the cost of faithfulness, making the quantifier diversity useless if the inferences are incorrect. GPT-OSS-120B exhibits the second lowest redundancy, while GPT-OSS-20B and LLAMA-70B rely more heavily on a smaller set of lexical realizations.

7 Human Evaluation

While the Python checker verification provides a scalable method to evaluate logical faithfulness,

Model	Python Checker (%)	Human Annotation (%)	Agreement (%)
LLaMA 3.1 8B	46.67%	56.33%	79.67%
LLaMA 3.1 70B	68.33%	75.33%	91.67%
GPT-OSS 20B	89.00%	92.33%	96.00%
GPT-OSS 120B	**94.33%**	**96.50%**	**97.50%**
Overall	74.58%	80.13%	**91.21%**

Table 3: Human evaluation random 30 tables per model. Python Checker reports the automatically computed inference accuracy, Human Annotation reports the manually annotated accuracy, and Agreement denotes the agreement between the automated and human judgments.

the scripts are generated by LLMs and therefore may contain implementation errors that misclassify inferences. To measure the reliability of STAT-TO-TEXT, we manually annotate the quantified inferences generated and verified for 30 randomly sampled tables of the 100 tables from all four models. This leads to 30 tables across 4 models, with a total of 120 tables annotated. Results are illustrated in Table 3. Overall, the automated verdicts agree with the human labels on 91.2%, where we find more false negatives in the results of the pipeline than false positives. Of the 211 disagreement cases, 172 were false negatives, where the verification script labeled a human judged true inference as false, while the remaining 39 were false positives. Therefore, the automated validity scores should be interpreted as conservative lower bounds on true inference validity. Consistent with our previous evaluation, scale and family does matter, once again with the highest agreement observed with GPT-OSS-120B, and the lowest in LLAMA-3.1-8B, both in inference generation and generating Python checker verification accuracy.

During this qualitative evaluation, we observed several unexpected failures, most concentrated with the LLAMA-3.1-8B. These failures extend to verification code execution as well as generated inferences. The first involves cases where the checker is able to select the correct rows that are relevant to an inference but then assigns an incorrect "True" or "False" label while listing those rows as supporting a violation (Appendix E). The second is related to LLMs performing a quantifier logical failure. In this case, majority/non-majority claims in which any non-satisfying row is counted as a violation, so "most" and "there exists at least" is evaluated as if it was "all" (Appendix E). A third recurring failure is a verdict that contradicts its own justification: in at least 8 of the disagreements the checker's explanation reports zero violating rows, yet the label it emits is still "false" (Appendix E). A fourth failure involves the checker's inability to correctly evaluate "X or Y" statements, instead marking the entire statement as "False" when one of the two conditions is not satisfied (Appendix E). Despite this, the human evaluation supports the conclusion that execution based verification with LLM generated code is a reliable evaluation instrument when the code generating model is sufficiently capable, reaching 91.5–97.5% agreement with human judgments for the three strongest models.

8 Discussion

RQ1. *How reliably does our* STAT-TO-TEXT *pipeline generate and verify quantified natural-language inferences from statistical tables?*

Findings: Our automated and human evaluation demonstrates that the STAT-TO-TEXT pipeline can reliably generate and verify quantified natural-language inferences from statistical tables, provided an appropriate model is selected. Among the evaluated models, GPT-OSS-120B performs best, generating faithful quantified inferences without sacrificing coverage or diversity. Furthermore, it achieves the highest agreement with human annotations (97.5%), demonstrating that its automatically verified inferences closely align with human judgments, concluding that our STAT-TO-TEXT pipeline is reliable for quantified inference generation from statistical tables.

RQ2. *How do model family, model scale, and reasoning paradigm (direct prediction vs. reasoning models) affect the logical faithfulness of quantified* STAT-TO-TEXT *generation?*

Findings: Our results show that model family, model scale, and, in this case, reasoning paradigm have the strongest influence on logical faithfulness, with GPT-OSS-120B consistently outperforming all other evaluated models. We also observe a

clear model family effect, as the GPT-OSS models produce substantially more faithful quantified inferences than the LLAMA models despite exhibiting lower quantifier diversity. The LLAMA models generate more diverse quantifier categories and achieve slightly broader variable coverage; however, these advantages do not translate into greater logical faithfulness. As quantified inferences must first be faithful before diversity becomes meaningful, the GPT-OSS models provide the strongest overall performance. As for the reasoning paradigm, further research is needed to isolate and definitively determine its contribution independently of model family and architecture.

RQ3. *How do LLMs differ in the coverage and diversity of the quantified inferences they generate?* **Findings:** Our results show that models do not struggle with coverage regardless of performance on other metrics. However, diversity of quantifiers seems to come at a cost for faithfulness, especially with smaller models. With the exception of GPT-OSS-120B, models with higher faithfulness seem to exhibit lower quantifier diversity and vice versa, which will be the focus of our future research.

9 Conclusions and Future Work

In this paper, we present a new pipeline, STAT-TO-TEXT, which reliably generates and verifies faithful quantified inferences from statistical tables. We find that while all models achieve high coverage of information from the source tables, model family and model scale have a substantial impact on the logical faithfulness of the generated inferences. Although the LLAMA models exhibit greater quantifier diversity and slightly broader variable coverage, the GPT-OSS models consistently produce more faithful quantified inferences. Finally, we find high agreement between the automated evaluation and human annotators, showing that the proposed pipeline provides a reliable framework for future quantified table-to-text generation.

Future work will focus on incorporating real world tables and improving the reliability of the Python verifier to better distinguish inference generation errors from verification errors, potentially through stronger models or supervised training on verification code. We also plan to evaluate the pipeline on larger and more complex statistical tables, where accurately verifying quantified statements becomes increasingly challenging. Finally, we aim to extend STAT-TO-TEXT to coherent multi-sentence discourse inferences while preserving executable verification of individual claims and discourse relations connecting them.

Limitations

Our experiments use small, simplified tables with controlled table schemas, so the findings may not generalize to larger, noisier, or more structurally complex real world tables. Given that the throughput of the pipeline is not 100%, some mismatched tables are dropped by different models during pipeline execution, which may impact the evaluation results. Because each model generates both the inferences and their verification code, the reported accuracy reflects the full pipeline and may be affected by verifier errors, as shown by the human evaluation. Finally, the predefined inference types and the confounding of model family, scale, and reasoning paradigm limit conclusions about unconstrained reasoning and the independent effects of these factors. Our findings should therefore be interpreted within the scope of the current experimental setting.

Acknowledgments

We thank Ryan Dolan for the valuable discussions and insights that enhanced this work, the Illinois Computational Linguistics Lab for their supportive feedback during the project's initial stages, and the anonymous reviewers for their helpful comments.

References

John Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219.

Eduardo Calò, Saad Mahamood, Albert Gatt, and Kees Van Deemter. 2026. A logic-based approach to hallucinations in data-to-text NLG: Experiments with human and LLM annotators. In *Proceedings of the 15th Joint Conference on Lexical and Computational Semantics (*SEM 2026)*, pages 28–62, San Diego, California, United States. Association for Computational Linguistics.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Guanyi Chen, Kees van Deemter, and Chenghua Lin. 2019. Generating quantified descriptions of abstract visual scenes. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 529–539, Tokyo, Japan. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020b. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2096–2111, Online. Association for Computational Linguistics.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Li Gong, Josep Crego, and Jean Senellart. 2019. Enhanced transformer model for data-to-text generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 148–156, Hong Kong. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zdeněk Kasner, Simon Mille, and Ondřej Dušek. 2021. Text-in-context: Token-level error detection for table-to-text generation. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 259–265, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Yupian Lin, Tong Ruan, Jingping Liu, and Haofen Wang. 2024. A survey on neural data-to-text generation. *IEEE Transactions on Knowledge and Data Engineering*, 36(4):1431–1449.

Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. 2022. PLOG: Table-to-logic pretraining for logical table-to-text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5531–5546, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tianyu Liu, Xin Zheng, Baobao Chang, and Zhifang Sui. 2021. Towards faithfulness in open domain table-to-text generation from an entity-centric view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13415–13423.

Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. 2021. SciGen: a dataset for reasoning-aware text generation from scientific tables. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Kristýna Onderková, Ondrej Platek, Zdeněk Kasner, and Ondrej Dusek. 2025. FreshTab: Sourcing fresh data for table-to-text generation evaluation. In *Proceedings of the 18th International Natural Language Generation Conference*, pages 108–121, Hanoi, Vietnam. Association for Computational Linguistics.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

OpenAI. 2026. GPT-5.5 System Card. OpenAI.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Alberto Sánchez Pérez, Alaa Boukhary, Paolo Papotti, Luis Castejón Lozano, and Adam Elwood. 2025. An LLM-based approach for insight generation in data analysis. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 562–582, Albuquerque, New Mexico. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, United Kingdom.

Linfeng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7987–7998, Online. Association for Computational Linguistics.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. 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, Online. Association for Computational Linguistics.

Lena Trigg and Dean F. Hougen. 2025. Logical table-to-text generation: Challenges, methods, and reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and*

the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1663–1677, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

Peng Wang, Junyang Lin, An Yang, Chang Zhou, Yichang Zhang, Jingren Zhou, and Hongxia Yang. 2021. Sketch and refine: Towards faithful and informative table-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4831–4843, Online. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Xunjian Yin and Xiaojun Wan. 2022. How do Seq2Seq models perform on end-to-end data-to-text generation? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7701–7710, Dublin, Ireland. Association for Computational Linguistics.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023. 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.

A LLM Inference Pipeline Full Prompts

A.1 Synthetic Table Generation Prompt

```
Generate exactly one synthetic statistical table in valid CSV format.

Requirements:
1. The table must have between 5 and 10 columns.
2. The table must contain exactly 15 data rows plus 1 header row.
3. Invent realistic column names, including:
   - numeric columns (integers and/or floats)
   - categorical columns
4. Numeric columns must contain only plain numbers
   (no $, commas, or % symbols).
5. Make the data plausible for a real-world dataset.
6. Ensure values vary across rows.
7. All rows must contain the same number of columns
   as the header.
8. Do not include commas inside cell values.
9. Do not include explanations, markdown, code fences,
   titles, or any text before or after the CSV.
10. Output only the CSV table.

The dataset may describe any realistic domain such as health, education, business,
↪  transportation, or demographics.
```

A.2 Natural-Language Inference Generation Prompt

```
Your task is to generate 20 natural language statements that describe patterns, relationships,
↪  and notable observations in this data.

REQUIREMENTS:
1. Factual Accuracy:
   Each statement must be verifiable against the actual data. Avoid generalizations that
   ↪  contradict even a single record.

2. Semantic Salience:
   Focus on relationships between variables that are semantically meaningful (e.g., "women aged
   ↪  21-43" or "high-income individuals"). Avoid arbitrary correlations.

3. Clarity:
   Use natural language that is easy to understand. Avoid overly complex nested conditions.

4. Variety:
   - Universal claims:
       "All X satisfy property Y"
   - Existential claims:
       "There exists at least one X that satisfies Y"
   - Majority/frequency claims:
       "Most X have property Y"
   - Negation:
       "No X where Y ..." (check subgroup max/min first)
   - Conditional claims:
       "If X, then Y"

AVOID:
- Overly specific conditions that apply to only 1-2 individuals.
- Redundant statements expressing the same fact in different ways.
- Statements that are too complex to parse (keep conditions to 2-3 attributes).
- Probabilistic language (e.g., "likely", "probably")
  unless strongly supported by the data.
```

A.3 Python Verification Prompt: LLAMA-3.1-8B

```
Do NOT repeat the instructions or the code provided.
Only output the requested Python code. Do NOT wrap the code in markdown fences.
Generate a complete standalone Python script.

Create exactly one function for every input statement, named sequentially:
stmt_1, stmt_2, stmt_3, ...

The checks list must include exactly the same functions:
checks = [(1, stmt_1), (2, stmt_2), ...]

Given the statements {lines} and the table headers {df.head(0)}, write Python code using pandas
↪  to check whether each statement is True or False and print a justification.

The CSV is already located at "{full_csv_path}". Hardcode this path
directly in the script.

Convert all numeric-looking columns to integers or floats using:
pd.to_numeric(..., errors="coerce").

Each statement function should explicitly convert any numeric columns it uses before checking
↪  conditions.

Truth must always be a single Python boolean, never a pandas Series. Use .any() or .all() before
↪  bool(...) whenever an expression could return a Series.

For implication statements of the form "if A then B", always use:

antecedent = ...
consequent = ...
viol = local[antecedent & ~(consequent)]
truth = bool(viol.empty)

For boolean masks involving multiple comparisons, wrap each comparison
in parentheses:

mask = (local["x"] >= 13) & (local["y"] < 54)

Never write:

mask = local["x"] >= 13 & local["y"] < 54

Use .isin(...) directly on a Series, not on .str.

When selecting dataframe columns, always use a list, never a set.

Do not put dataframe indexing expressions directly inside f-strings.
Assign them first:

preview = viol[needed].to_dict(orient="records")
expl = f"Found {len(viol)} violating rows. Violating rows: {preview}."

Do not use numeric comparisons on categorical yes/no columns unless
the column has first been safely normalized.

Distinguish column names from category values. If a medical condition,
diagnosis, treatment type, category, group, region, gender, weather type,
or similar label appears in a statement but is not an exact column name,
do not invent a new column for it. Instead, identify the relevant
categorical column and compare against normalized string values.

For yes/no columns such as smoker, diabetes, asthma, hypertension,
or club_member, normalize the column first:

col_norm = local["smoker"].astype(str).str.strip().str.lower()
smoker_yes = col_norm.isin(["yes", "y", "true", "t", "1"])

Everything output must be valid, immediately runnable Python with no
markdown or commentary outside comments.
```

A.4 Initial Python Verification Prompt: GPT-OSS

```
System:
You are an expert data analyst.

User:
Do NOT repeat the instructions or the code provided.
Only output the requested Python code. Do NOT wrap the code
in markdown fences.

Given the following statements:

{statements_text}

and the following CSV preview showing the header row:

{table_head}

write Python code using pandas that checks whether each
statement is True or False and prints a justification.

The CSV is already located at:
"{csv_path}"

Hardcode this path directly in the script.

Requirements:
- Use pandas.
- Read the CSV from the hardcoded path.
- Convert numeric columns stored as strings into numeric values.
- Convert turn-number columns to integers when appropriate.
- Check every statement.
- Print whether each statement is True or False.
- Print a short justification for each result.
- Output only valid, immediately runnable Python.
- Do not include markdown or commentary outside Python comments.

An example Python script illustrating the expected output
format was appended to the prompt.
```

B Synthetic Tables

patient_id	diagnosis	age	bp_systolic	bp_diastolic	cholesterol_mg_dl	bmi	smoker
PT001001	asthma	46	129	72	192	33.3	no
PT001002	diabetes	62	143	82	223	22.5	yes
PT001003	diabetes	68	147	87	198	31.0	no
PT001004	arthritis	46	135	77	182	27.6	yes
PT001005	hypertension	76	119	74	245	22.7	no
PT001006	arthritis	25	136	82	241	34.4	no
PT001007	arthritis	76	112	91	179	30.0	no
PT001008	asthma	28	130	83	185	26.9	no
PT001009	arthritis	69	123	86	178	32.7	no
PT001010	arthritis	59	124	74	212	31.2	yes
PT001011	arthritis	41	143	70	179	33.5	no
PT001012	diabetes	24	127	98	237	33.8	yes
PT001013	migraine	73	116	94	233	31.2	yes
PT001014	migraine	56	122	78	232	32.7	no
PT001015	diabetes	55	156	76	204	26.1	no

Table 4: Example of a synthetic statistical table generated by GPT-5.5

C Example Generated Inferences and Python Checker Output

C.1 Generated Natural-Language Inferences

The following quantified inferences were generated by GPT-OSS-120B from the synthetic patient table shown in Table 4.

```
1.  All asthma patients are non-smokers.
2.  Most arthritis patients are non-smokers.
3.  There exists at least one diabetes patient who is a smoker.
4.  No hypertension patient is a smoker.
5.  If a patient is a smoker, their cholesterol level is at least 182 mg/dL.
6.  All patients with BMI greater than 33 have cholesterol of at least 179 mg/dL.
7.  No patient with cholesterol below 180 mg/dL is a smoker.
8.  All patients aged 70 or older have cholesterol above 230 mg/dL.
9.  All patients with systolic blood pressure above 140 have either diabetes or arthritis.
10. There exists at least one patient with BMI between 20 and 25 who is a smoker.
11. No patient with BMI below 25 has diastolic blood pressure above 90.
12. All patients with cholesterol above 240 mg/dL are non-smokers.
13. Most patients with BMI over 30 are non-smokers.
14. If a patient is diagnosed with migraine, their systolic blood pressure is below 130.
15. All patients with diastolic blood pressure of 70 are diagnosed with arthritis.
16. No smoker has a BMI below 22.5.
17. Most patients older than 60 have cholesterol above 200 mg/dL.
18. All patients with diastolic blood pressure of 91 are non-smokers.
19. If a patient is non-smoker, their diastolic blood pressure is at most 91.
20. All patients with cholesterol exactly 179 mg/dL are non-smokers.
```

C.2 Python Checker Output

The following output was generated by GPT-OSS-120B when verifying the quantified inferences shown above against the synthetic patient table in Table 4.

```
Statement 1: TRUE
  - 1. All asthma patients are non-smokers.
  - Explanation: All 2 asthma patients are non-smokers.

Statement 2: TRUE
  - 2. Most arthritis patients are non-smokers.
  - Explanation: 66.7% of 6 arthritis patients are non-smokers.

Statement 3: TRUE
  - 3. There exists at least one diabetes patient who is a smoker.
  - Explanation: Found 2 diabetes patient(s) who smoke.

Statement 4: TRUE
  - 4. No hypertension patient is a smoker.
  - Explanation: All hypertension patients are non-smokers.

Statement 5: TRUE
  - 5. If a patient is a smoker, their cholesterol level is at least 182 mg/dL.
  - Explanation: All 5 smokers have cholesterol >=182.

Statement 6: TRUE
  - 6. All patients with BMI greater than 33 have cholesterol of at least 179 mg/dL.
  - Explanation: All 4 high-BMI patients meet the cholesterol threshold.

Statement 7: TRUE
  - 7. No patient with cholesterol below 180 mg/dL is a smoker.
  - Explanation: All low-cholesterol patients are non-smokers.

Statement 8: FALSE
  - 8. All patients aged 70 or older have cholesterol above 230 mg/dL.
  - Explanation: 1 senior patient(s) have cholesterol <=230 (ids: PT001007).
```

```
Statement 9: TRUE
  - 9. All patients with systolic blood pressure above 140 have either diabetes or arthritis.
  - Explanation: All 4 high-systolic patients have diabetes or arthritis.

Statement 10: TRUE
  - 10. There exists at least one patient with BMI between 20 and 25 who is a smoker.
  - Explanation: Found 1 smoker(s) with BMI 20-25.

Statement 11: TRUE
  - 11. No patient with BMI below 25 has diastolic blood pressure above 90.
  - Explanation: All 2 low-BMI patients have diastolic <=90.

Statement 12: TRUE
  - 12. All patients with cholesterol above 240 mg/dL are non-smokers.
  - Explanation: All 2 high-cholesterol patients are non-smokers.

Statement 13: TRUE
  - 13. Most patients with BMI over 30 are non-smokers.
  - Explanation: 66.7% of 9 BMI>30 patients are non-smokers.

Statement 14: TRUE
  - 14. If a patient is diagnosed with migraine, their systolic blood pressure is below 130.
  - Explanation: All 2 migraine patients have systolic <130.

Statement 15: TRUE
  - 15. All patients with diastolic blood pressure of 70 are diagnosed with arthritis.
  - Explanation: All 1 patients with DBP=70 have arthritis.

Statement 16: TRUE
  - 16. No smoker has a BMI below 22.5.
  - Explanation: All 5 smokers have BMI >=22.5.

Statement 17: FALSE
  - 17. Most patients older than 60 have cholesterol above 200 mg/dL.
  - Explanation: 50.0% of 6 patients >60 have cholesterol >200.

Statement 18: TRUE
  - 18. All patients with diastolic blood pressure of 91 are non-smokers.
  - Explanation: All 1 patients with DBP=91 are non-smokers.

Statement 19: TRUE
  - 19. If a patient is non-smoker, their diastolic blood pressure is at most 91.
  - Explanation: All 10 non-smokers have diastolic <=91.

Statement 20: TRUE
  - 20. All patients with cholesterol exactly 179 mg/dL are non-smokers.
  - Explanation: All 2 patients with cholesterol 179 are non-smokers.
```

D Metrics

	LLM Inferences BEFORE Python Checker							
	# Tables	**Grounding↑**	**Accuracy↑**	**TRC↑**	**VC↑**	**VR↓**	**QC↑**	**QR↓**
LLAMA-8B	100	**1**	-	**1**	0.947	0.184	0.99	**0.769**
LLAMA-70B	100	**1**	-	**1**	0.956	0.169	**1**	0.941
GPT-OSS-20B	91	0.998	-	0.982	**0.958**	**0.142**	0.798	0.937
GPT-OSS-120B	100	0.995	-	**1**	0.955	0.165	0.96	0.818

Table 5: Performance of open-weight LLMs on STAT-TO-TEXT generated inferences' grounding, coverage (TRC, VC, QC), and diversity (VR, QR) *before* Python-based verification. Accuracy is non-existent before Python inferences checker. Metrics annotated with ↑ indicate that higher values correspond to better performance, whereas metrics annotated with ↓ indicate that lower values are preferred.

E Examples of Disagreements Between the Python Checker and Human Annotation

Example 1: Correct Row Selection with an Incorrect Verdict

Model: LLAMA-3.1-8B

Statement. There exists at least one farm that grows corn and has a fertilizer application of less than 645 kg.

Checker result: FALSE
Human label: TRUE

Checker explanation. `Found 1 violating rows. Violating rows: [{'crop_type': 'corn', 'fertilizer_kg': 611}].`

Error analysis. The checker identifies a row containing a corn farm with a fertilizer application of 611 kg, which satisfies the existential claim. However, it incorrectly treats this satisfying row as a violation and assigns a false verdict.

Example 2: Majority Quantifier Evaluated as Universal

Model: LLAMA-3.1-8B

Statement. Most employees have a performance rating of 4.0 or higher.

Checker result: FALSE
Human label: TRUE

Checker explanation. `Found 1 violating rows. Violating rows: [{'employee_id': 'E019002', 'performance_rating': 3.8}].`

Error analysis. Only one employee has a performance rating below 4.0, so the majority of employees satisfy the stated threshold. The checker incorrectly rejects the statement because it finds one non-satisfying employee, effectively evaluating *most* as if it were the universal quantifier *all*.

Example 3: Verdict Contradicting the Checker Explanation

Model: LLAMA-3.1-8B

Statement. For all individuals with a diagnosis of hypertension, their age is greater than 40.

Checker result: FALSE
Human label: TRUE

Checker explanation. `Found 0 violating rows. Violating rows: [].`

Error analysis. The checker reports that no violating rows were found, which supports the universal statement. Nevertheless, it outputs a false verdict, directly contradicting its own explanation.

Example 4: Disjunction Evaluated Incorrectly

Model: LLAMA-3.1-8B

Statement. All players in the table are either guards, centers, or forwards.

Checker result: FALSE
Human label: TRUE

Checker explanation.

```
Found 15 violating rows. Violating rows: [{'position': 'guard'}, {'position': 'guard'},
{'position': 'guard'}, {'position': 'center'}, {'position': 'center'}, {'position': 'forward'},
{'position': 'forward'}, {'position': 'guard'}, {'position': 'forward'}, {'position': 'forward'},
{'position': 'guard'}, {'position': 'guard'}, {'position': 'guard'}, {'position': 'forward'},
{'position': 'center'}].
```

Error analysis. Every row contains one of the three permitted position labels—`guard`, `center`, or `forward`—so no row violates the disjunctive condition. Nevertheless, the checker marks all 15 rows as violations and assigns a false verdict.