Finding Support for Tabular LLM Outputs

Grace Fan
Northeastern University
Boston, United States
fan.gr@northeastern.edu

Roee Shraga
Worcester Polytechnic Institute
Worcester, United States
rshraga@wpi.edu

Renée J. Miller
Northeastern U. & U. Waterloo
Boston, United States
miller@northeastern.edu

ABSTRACT

With the emerging advancements of AI, validating data generated by AI models becomes a key challenge. In this work, we tackle the problem of validating tabular data generated by large language models (LLMs). By leveraging a recently proposed technique called Gen-T, we present a technique to verify if the data in the LLM table can be reclaimed (reproduced) using tables available in a given data lake (for example, tables used to train the LLM). Specifically, we measure the number of data lake tables that support tuples (or partial tuples) in a generated table. We further provide suggestions for value replacements if a generated value cannot be reclaimed. Using this approach, users can evaluate their LLM-generated tables, consider potential modifications for table values, and gauge how much support the modified table has from the data lake. We discuss two case studies showing that table values annotated with reclamation support scores, along with possible value replacements, can help users assess the trustworthiness of LLM-generated tables.

VLDB Workshop Reference Format:
Grace Fan, Roee Shraga, and Renée J. Miller. Finding Support for Tabular LLM Outputs. VLDB 2024 Workshop: Tabular Data Analysis Workshop (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/northeastern-datalab/table-validation.

1 INTRODUCTION

Verifying the output of generative AI or large language models (LLMs) using a data management lens is an emerging and important area [14, 36]. For example, users who generate summary tables and charts (e.g., Microsoft Copilot [28]) or presentation slides (e.g., SlidesAI [35]) from input data would find it useful to verify model outputs and examine what data may have been used to generate them. Associating AI-generated data (or any automatically created data) with some form of evidence for validity will give users more confidence in the data they are using.

Many applications require the generation of tabular data including benchmarking where synthetic data may be used [37], or where real data can be modified systematically to generate new tables for benchmarking solutions to problems such as data cleaning [1]. Other applications have used knowledge graphs [9], Git repositories [20], and the web [6] to generate tabular data. Diffusion models have also been used to synthesize tabular data [25, 41]. Recently, Pal et al. proposed a method to generate tabular data using LLMs [32] to create benchmarks for semantic data management problems. These methods may require manual verification of the generated tables, which may provide users with an intuitive sense of what evidence is being used in the verification. However, manual verification is laborious. This is the problem we tackle: *how do we verify (LLM) generated tables in a clear and intuitive way that allows a user to easily assess what information is being used for the verification and the degree of support different tuples in a generated table may have.*

We build on top of a recently proposed technique called Gen-T [10] and present a technique to verify if the data in a generated table can be reclaimed (reproduced) using tables available in a given data lake (for example, tables used to train an LLM that generated the table). Given a table (referred to as a Source Table)[1], Gen-T searches the data lake to find a set of *supporting tables* that can be integrated to reclaim (reproduce) the generated table as accurately as possible. Given a reclaimed table, first we show the degree of support for tuples (or partial tuples) from the supporting tables and allow the user to examine the supporting tables and second, we suggest corrections (value replacements) for values that cannot be reclaimed.

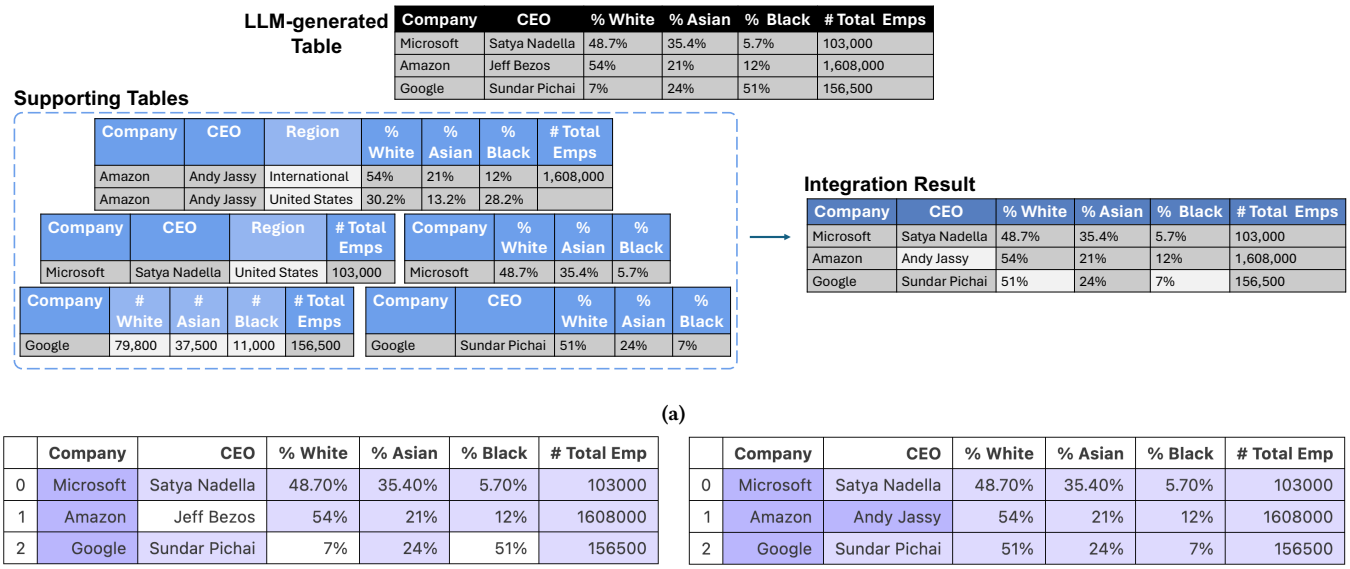
1.1 Motivating Example

Assume a data scientist wants to analyze diversity reports of different high-tech companies. To do so, the scientist uses (prompts) an LLM to "show demographics of employees in Top US tech companies in 2021". Figure 1a (top table) shows such an LLM-generated[2] table including companies, their CEOs, and their employee demographics. In addition, assume that the scientist has access to a data lake. The bottom left side of Figure 1a shows a subset of data lake tables, which we refer to as supporting tables. These tables are considered as possible tables that the LLM used to generate its table.

Our proposed framework first filters the data lake to generate a set of supporting tables using Gen-T [10]. Our solution computes which supporting tables support which portions of the reclaimed table and using this, a scientist can already detect that row information about 'Amazon' in the LLM table are supported by a table containing 'International' statistics, whereas statistics reported for 'Microsoft' in the reclaimed table supported by tuples reporting US statistics. Using this supporting information, users can verify the LLM values. Our method automates this effort by computing *support scores* for the source table values. A value that cannot be reclaimed has a support score of zero (and would appear as a null in Gen-T's output). In this simple example, the support scores are

Proceedings of the VLDB Endowment. ISSN 2150-8097.

[1] Which, in this work, we assume is LLM-generated.
[2] In the example, we used the free research preview of GPT 3.5.



(a)

	Company	CEO	% White	% Asian	% Black	# Total Emp
0	Microsoft	Satya Nadella	48.70%	35.40%	5.70%	103000
1	Amazon	Jeff Bezos	54%	21%	12%	1608000
2	Google	Sundar Pichai	7%	24%	51%	156500

(b) LLM-generated table

	Company	CEO	% White	% Asian	% Black	# Total Emp
0	Microsoft	Satya Nadella	48.70%	35.40%	5.70%	103000
1	Amazon	Andy Jassy	54%	21%	12%	1608000
2	Google	Sundar Pichai	51%	24%	7%	156500

(c) Verified table

Figure 1: (a) LLM-generated table (top) and supporting tables (bottom left) that the LLM may have used to produce the top table. By integrating the supporting tables, Gen-T produces a reclaimed table (not shown) that has nulls for three values. Our method adds to the reclaimed table support scores (b) and suggests values for the unreclaimed values to generate a possible Verified (and Corrected) Table (bottom right in (a)). The LLM-generated Table and verified table from (a) are annotated with support scores in (b), (c), respectively. Every cell is color-coded according to its support score, with darker cells indicating higher scores and lighter shades indicating lower scores.

all one or two except for the three values that cannot be reclaimed. Our *support and verification* method suggests corrections for values with zero support scores. A possible verified table is presented in the bottom right side of Figure 1a, in which the `%White` and `%Black` values in the third row are corrected and the `CEO` is corrected to `Andy Jassy`.

1.2 Contributions

In this work, we present a framework to verify and correct tabular data generated by LLMs (or any automated method). In summary, we make the following contributions.

(1) We measure the degree of support that a generated table has from a set of supporting tables that can be used to reclaim the table as accurately as possible.
(2) We suggest replacements for values that cannot be reclaimed (or that a user questions, perhaps due to their low support). We use Gen-T to compute a value that would have the highest support from supporting tables and suggest this as a way to correct or repair the generated table.
(3) We conduct two case studies to show the value of measuring support and using this to verify the tabular output of LLMs and to suggest possible corrections for values that cannot be reclaimed. We show that our method out-performs a recent proposal by Narayan et al. [29] for using LLMs to correct data.

All our code and data is open source.[3]

[3]https://github.com/northeastern-datalab/table-validation

2 RELATED WORK

In our framework, we first search for a set of data lake tables that support a given source table. From this set of tables, we can suggest replacements for values in the source table that have low support.

2.1 Table Discovery and Query-by-Example

Our work uses Gen-T for supporting table discovery and reclamation [10]. Table discovery is a long-standing problem [11] that can be solved using keyword search over tables, unionable table search, joinable table search, or various forms of related table search. Early work such as Octopus [7] and Google Dataset Search [5] support keyword search over the metadata of tables [26] and smaller scale web-tables [38]. Data-driven table discovery systems [15, 30, 33, 42, 43] were then developed to find schema complements, entity complements, joinable tables, and unionable tables. Recent work relies on value overlap between the columns [3, 42, 43], knowledge graph concept or value embedding similarity [23, 30], and similarity of table representations [12, 19]. Some table discovery methods [13] capture relationships between tables by building an enterprise knowledge graph, but only aim to solve the table discovery problem and do not perform table integration or reclamation. Other recent work [16] is also goal-oriented discovery for specific downstream tasks, aiming to augment columns. Gen-T uses table discovery then performs reclamation of a given Source Table.

Query-by-Example methods [18] also aim to reproduce an input table, but with the goal of completing a partial input table. Unlike these methods, Gen-T aims to reproduce the input table as closely

as possible, making no assumption that the input table is only partial. Gen-T uses more integration operators than most QBE methods, namely Select-Project-Join-Union queries, since it aims to integrate tables to do the reclamation. Gen-T is also similar to Query-by-Target methods [40] that synthesize a pipeline to create a target table from a given set of tables. However, Gen-T does not assume the set of tables or integration query that first generated a given table are known. This aligns with our framework, since we do not know what tables an LLM may have used to produce a tabular output. Finally, most QBE approaches assume a small example table while Gen-T can reclaim large tables with thousands of tuples.

2.2 Error Correction

Another line of related work is error correction. Error correction may use knowledge bases and crowdsourcing [8], or reference data [8], among other approaches. An important line of work uses the notion of *minimal repair* to suggest corrections that would require the fewest number of changes [21]. Our maximum support corrections are motivated by the intuition that tables in a data lake each provide independent evidence for a repair so we use the correction with the highest evidence. An interesting line of future work would be to consider whether supporting tables are in fact independent or whether they are versions of each other [34].

3 METHOD OVERVIEW

We propose a verification framework that first reclaims a source table, and then computes support values and uses these to suggest corrections for values that cannot be reclaimed.

3.1 Preliminaries

To verify the values of a source table generated from an LLM, we first find tables from a data lake that support and reclaim the source table. We do not assume that we know the exact set of tables that the LLM used to first generate the source table, especially a closed-source LLM, so we do not know if we can find a set of data lake tables that reproduces the entire source table. To search for data lake tables that can support the source table, we need to compare columns and tuples in data lake tables to those in the source table. Thus, we first perform schema matching and instance comparison. Following Gen-T [10], we perform schema matching by finding tables with columns that have high value overlap with columns from the source table, and rename the tables' columns with the source table's column name. To efficiently find data lake tables that can reproduce the source table, we follow Gen-T and make use of key column(s) in the source table. While Gen-T does not assume data lake tables contain keys or have foreign key relationships, Gen-T does require the source table to have a key so that instance (table) comparison is fast [10] (as instance comparison is done often in their solution).

3.2 Reclaiming LLM Tables

We provide Gen-T with a (LLM-generated) source table, along with a data lake (which may include set of tables used to train the LLM). Gen-T discovers a set of *supporting tables* that are likely to be tables from which the LLM-generated data may have originated, and thus support values in the source table. Gen-T integrates the supporting tables in a way that reclaims as many of the source values as possible, outputting a reclaimed table. The reclaimed table is identical to the source table if all values can be reclaimed, otherwise it has nulls for values that cannot be reclaimed.

3.3 Degree of Support

To verify a source table, we first compute support values for all values in the reclaimed table. These support values give users further insight into the source table by evaluating how often the table's values are found in supporting tables. To accurately find the support that each value has, we use the source key and search the supporting tables for all occurrences of the source key and key-value pairs. Thus, we find support scores by encoding the degree of support that each key and key-value pair has.

Example 1. *In Figure 1, recall that the value* `Jeff Bezos` *cannot be reclaimed from the given supporting tables. The key-value pair* (`Microsoft, Satya Nadella`) *has support 1, while* (`Amazon, Andy Jassy`) *has support 2.*

3.4 Suggesting Value Replacements

If values in the source table cannot be reclaimed or have low support, we suggest possible replacements for these values. Thus, we use support values to suggest corrections to a source table. If the supporting tables have values with higher support scores than unreclaimed source values, these values can be suggested as possible replacements to the unreclaimed source values. We take values with the highest support from supporting tables, and suggest them as corrections to the source table. The user can then choose to replace source table's values with suggested values. Note that we do not assume that suggested value replacements from supporting tables are correct, we only provide one method for suggesting replacements for the source table. While there are other ways to evaluate and compare degrees of support, such as trustworthiness, completeness, and accuracy of supporting tables, we leave this to future work.

Example 2. *In Figure 1, Figure 1b shows the source table and Figure 1c shows the verified table from Figure 1a, both annotated with support scores. Cells are color-coded to show the degree of support that each table value has, with darker shades indicating a high support and lighter shades indicating a lower. Comparing the two heatmaps of support scores, we notice that the source table 1b has no support for some Google values (in white), whereas the verified table 1c has different Google values with higher support (in blue). Thus, users can choose whether to replace values in their source table with high support values suggested by our verification and correction method.*

4 CASE STUDIES

We present two case studies in which we verify LLM-generated source tables and assess how much support they have from a set of tables known to be part of the LLM's training data. Note that these are preliminary studies intended to analyze LLM-generated tabular output using our methodology and provide insights to inspire future directions. In future work, we will develop evaluation metrics and

carry out extensive experiments to study this problem. We use Gen-T to reclaim as many values from the source tables as possible from the training data, and evaluate their support scores. For values that we cannot reclaim, we will illustrate what values our method suggests as replacements. The first case study is a controlled study (Section 4.1), in which we use an LLM [31] to generate source tables using tables from the TPC-H benchmark [37], and assess how much support the (LLM-generated) source tables have. In the second study (Section 4.2), we take a benchmark with tables previously generated from an LLM [32], [4] and analyze their support from an open data lake WikiTables [2] known to be included in the LLM's training data.

4.1 TPC-H Controlled Study

We start with eight tables from the TPC-H benchmark [37] with information on customer orders, suppliers, nations, etc. Using an LLM (ChatGPT3.5 [31]), we integrate different tables from the TPC-H benchmark to produce a tabular result. First, we serialize each TPC-H table into a string by concatenating cell values from each row. For table integration, we ask the LLM to consider Select-Project-Join-Union queries. We prompt the LLM to output a table that has a key column. Note that we consider a single key column in this controlled study, but our method can also handle multi-attribute keys. Figure 2 shows an example prompt we give to ChatGPT.

```
Given the following tables, integrate them using some
Select-Project-Join-Union query. The integration result
should have one key column, meaning it has all unique
values and no null value. Return the tabular result with
name nation_supplier_part_partsupp.csv separated by
semicolons. Ensure that all rows in the tabular result
have the same number of columns.
<serialized Table 1>
<serialized Table 2>
…
```

Figure 2: Sample LLM prompt to produce source tables. The LLM is prompted to generate a table from a set of TPC-H tables.

We analyze four tables generated by the LLM (our source tables). We then use Gen-T [10] to reclaim each table, and see if there are unreclaimed values (which would be the result of errors or hallucinations by the LLM) and whether any unreclaimed values can be correctly repaired by our method.

```
You are a scientist who detects and corrects errors in
tabular data. Return the corrected table, along with an
explanation of what errors you detected, and how you
fixed them using the provided data lake. Here is the
table you need to correct:
<serialized table>
Here is the data lake that you can use to correct values
in the table:
<serialized Table 1>
<serialized Table 2>
…
```

Figure 3: An LLM prompt to correct the source table using TPC-H tables, and return an LLM-corrected table. This serves as a baseline in our analyses.

[4]These tables were generated as a benchmark for the table union search problem.

To evaluate how well our method suggests value corrections, we also consider a baseline inspired by Narayan et al. [29], in which we use the same LLM to "correct" errors in the source table. We serialize the source table, along with TPC-H tables, and prompt the LLM to detect errors in the source table. Then, the LLM is prompted to correct these errors using the TPC-H tables and return the corrected table. It has been shown that an LLM's performance improves when it explains its thought process [39], so we also prompt the LLM to explain the errors it detected and how it fixed them. Figure 3 shows an example prompt to produce an LLM-corrected table.

4.1.1 Column-level analysis. We compare the support that source tables, verified tables, and LLM-corrected tables have from a data lake (TPC-H tables), shown in Table 1. In the last column of Table 1, we find how many columns (attributes) in the source table contain values that have some support. We compare this to the column-level support that the verified table and LLM-corrected table have.

Across all four source tables generated by an LLM, 64-90% of their columns have some support. In other words, 10-36% of columns have no support from the data lake. On the other hand, all columns in the verified tables have some support from the data lake. This shows that table reclamation is useful when validating values of a source table and guaranteeing support from a data lake.

Compared to the LLM-corrected table (our baseline), the verified table still produces more values supported by the data lake. By prompting an LLM to use a data lake to correct the source table, we find that its output values are still not always consistent with (supported by) the data lake. Only 50-80% of the tables' columns have some support, but 20-50% of their columns have no support from the data lake. Thus, our data-driven approach to finding value corrections for source tables is more accurate.

4.1.2 Row-level analysis. We perform a row-level analysis in which we see how many rows are fully supported by TPC-H tables. We find that no row in the source tables or the LLM-Corrected tables is fully supported by TPC-H tables. That is, no row has support for all values. By contrast, most if not all rows in the verified tables have support for all their values by the data lake.

Table 2 shows an analysis of partial data lake support for each row in source tables, verified tables, and LLM-corrected tables. Verified tables contain as many, if not more supported values than the source tables. Thus, users can use supported values from the verified tables to replace source table's values.

Example 3. *Figure 4 shows a source table (Figure 4a), Verified table (Figure 4b), and LLM-corrected table (Figure 4c). All cells are color-coded to show each value's degree of support from the TPC-H tables (darker colors indicate higher support). Each cell lists the supporting table(s) for that value, showing the TPC-H table(s) that support each value. Note that this is a controlled study in which each value has only a few supporting tables (1-2), but this is not generally the case. In Figure 4a, the source table does not have any support for any value in the "REGIONKEY" column and for some values in the "S_COMMENT" column. This implies that the LLM found wrong region key values for every nation. After reclaiming the source table using the TPC-H tables (Figure 4b), we find different region key values and supplier comments for every nation, that all*

Table Name	% Columns with support for some value		
	Source Table	Verified Table	LLM-Corrected Table
region_nation_supplier_1	80%	100%	80%
region_nation_customer_0	64%	100%	73%
region_nation_supplier_2	70%	100%	60%
nation_supplier_part_partsupp_2	90%	100%	50%

Table 1: Percentage of columns in each Source table, Verified table, and LLM-corrected table (baseline) that have some support from the data lake for some of their values.

Table Name	Method	# Rows	Supported Values / Row
region_nation_supplier_1	Source Table	5	6-8
	Verified Table	5	7-8
	LLM-Corrected Table	5	6-8
region_nation_customer_0	Source Table	5	6-7
	Verified Table	5	8
	LLM-Corrected Table	5	7-8
region_nation_supplier_2	Source Table	4	6-7
	Verified Table	4	8
	LLM-Corrected Table	4	2-5
nation_supplier_part_partsupp_2	Source Table	19	9
	Verified Table	19	9
	LLM-Corrected Table	19	2-4

Table 2: Number of rows and supporting values per row, for Source tables, Verified tables, and LLM-corrected tables.

have support from the data lake. We can refer to the listed supporting tables for values in the "REGIONKEY" column ("nation.csv") and further validate the values in the verified table.

In the LLM-corrected table (Figure 4c), values for columns "S_ACCTBAL", "S_PHONE", ..., "S_COMMENT" do appear in the TPC-H tables, but in rows with different "NATIONKEY" and "SUPPKEY" values. For example, values in the second tuple (1432.69, 18-179...) appear in the "supplier" table, but in a tuple where "NATIONKEY" is 8 and "SUPPKEY" is 12.

LLM-corrected tables have the same number of columns and rows as the source table. Note that a verified table may have fewer rows if the key value of the row does not appear in the data lake. However, an LLM may generate value corrections that have the same or less support from the data lake as the original value. As shown in Example 3 and Figure 4, an LLM can also falsely detect supported values as errors and replace them with values that have no support. New values in the LLM-corrected table may appear in different tuples with different key values, creating misaligned key-value pairs. This again shows that a data-driven approach for table verification is necessary to validate tabular values and suggest values with data lake support.

4.2 General LLM Study

In our second case study, we take an existing benchmark called UGen [32], which contains tables generated by an LLM (Mixtral-8x7B-Instruct [22]). We do not know what tables were used to create UGen's tables, so we take the WikiTable benchmark [2] and see if any of UGen's tables can be verified using the 540,000 WikiTables as a data lake.

Table Name	Source Table	Verified Table
Criminology	46.154%	64.103%
Sports	35.915%	57.576%
Climatology	28.785%	56.503%

Table 3: Percentage of values with support from the data lake

We present an analysis of three source tables from the UGen benchmark that have the most support (*"Criminology", "Sports", "Climatology"*) from the data lake. We use table reclamation to reclaim each table and produce verified tables of each source table. In regard to the LLM-corrected baseline, unlike our previous controlled study, we do not know *what* or *if* tables from the WikiTables benchmark generated the UGen tables. Prompting an LLM with the entire WikiTables benchmark is not realistic due to the token limit

	NATIONKEY	S_ACCTBAL	S_PHONE	S_ADDRESS	REGIONKEY	S_NAME	S_COMMENT	SUPPKEY
0	22 nation.csv supplier.csv	6565.11 supplier.csv	32-698-298... supplier.csv	1Y5lwEgpe3... supplier.csv	2	Supplier#0... supplier.csv	requests ...	42 supplier.csv
1	12 nation.csv supplier.csv	7773.41 supplier.csv	22-421-568... supplier.csv	Z5mLuAoTUE... supplier.csv	3	Supplier#0... supplier.csv	ously. fin...	43 supplier.csv
2	7 nation.csv supplier.csv	9759.38 supplier.csv	17-713-930... supplier.csv	kERxlLDnlI... supplier.csv	1	Supplier#0... supplier.csv	x. careful... supplier.csv	44 supplier.csv
3	9 nation.csv supplier.csv	2944.23 supplier.csv	19-189-635... supplier.csv	LcKnsa8XGt... supplier.csv	4	Supplier#0... supplier.csv	iously acc... supplier.csv	45 supplier.csv

(a) Source table

	NATIONKEY	S_ACCTBAL	S_PHONE	S_ADDRESS	REGIONKEY	S_NAME	S_COMMENT	SUPPKEY
0	22 nation.csv supplier.csv	6565.11 supplier.csv	32-698-298... supplier.csv	1Y5lwEgpe3... supplier.csv	3 nation.csv	Supplier#0... supplier.csv	fluffily ... supplier.csv	42 supplier.csv
1	12 nation.csv supplier.csv	7773.41 supplier.csv	22-421-568... supplier.csv	Z5mLuAoTUE... supplier.csv	2 nation.csv	Supplier#0... supplier.csv	unts. unus... supplier.csv	43 supplier.csv
2	7 nation.csv supplier.csv	9759.38 supplier.csv	17-713-930... supplier.csv	kERxlLDnlI... supplier.csv	3 nation.csv	Supplier#0... supplier.csv	x. careful... supplier.csv	44 supplier.csv
3	9 nation.csv supplier.csv	2944.23 supplier.csv	19-189-635... supplier.csv	LcKnsa8XGt... supplier.csv	2 nation.csv	Supplier#0... supplier.csv	iously acc... supplier.csv	45 supplier.csv

(b) Verified table

	NATIONKEY	S_ACCTBAL	S_PHONE	S_ADDRESS	REGIONKEY	S_NAME	S_COMMENT	SUPPKEY
0	22 nation.csv supplier.csv	2972.26 supplier.csv	32-822-502... supplier.csv	c2d,ESHRSk...	2	Supplier#0... supplier.csv	ously expr... supplier.csv	42 supplier.csv
1	12 nation.csv supplier.csv	1432.69	18-179-925...	aLIW q0HY...	3	Supplier#0...	al package...	43 supplier.csv
2	7 nation.csv supplier.csv	4641.08	25-843-787...	Bk7ah4CK8S...	1	Supplier#0...	riously ev...	44 supplier.csv
3	9 nation.csv supplier.csv	9107.22	13-727-620...	HK71HQyWoq...	4	Supplier#0...	requests e...	45 supplier.csv

(c) LLM-Corrected table

Figure 4: (a) LLM-generated source table from regions, nations, and supplier tables from the TPC-H benchmark, (b) the Verified table produced by Gen-T with value replacements, and (c) the LLM-corrected table produced by an LLM. Each cell has a value (in bold) and a list of supporting tables for that value (in red).

constraint of LLMs, so we cannot use an LLM to correct the tables using WikiTables as a data lake.

As a general study, we analyze how many values from each table have any support from the WikiTables data lake. Table 3 shows the percentage of values that have data lake support for each source table and their verified table. While some values in the source tables have support from the data lake (29-46% of values), most values do not have any support (54-71% of values). On the other hand, the verified tables have many more values with data lake support. This shows that our data-driven approach does provide valid value replacements.

To better understand the difference between values in a source table from the UGen benchmark and a Verified table, we analyze values from the "Criminology" table in Figure 5.

Example 4. *Figure 5a shows a subset of rows and columns from a table from the UGen benchmark about criminology reports, and Figure 5b shows its verified table. The first row in Figure 5a is fully*

	Incident Number	Victim Race	Victim Age	Victim Gender	Suspect Age
0	29 (23)	Black (1)	25 (1)	Male (2)	27 (1)
1	35 (22)	Asian (0)	28 (0)	Male (0)	nan (0)
2	40 (18)	Asian (0)	25 (0)	Female (0)	nan (0)

(a) Source table

	Incident Number	Victim Race	Victim Age	Victim Gender	Suspect Age
0	29 (23)	Black (1)	25 (1)	Male (2)	27 (1)
1	35 (22)	Black (1)	28 (0)	Male (0)	nan (0)
2	40 (18)	White (1)	47 (0)	Female (0)	nan (0)

(b) Verified table

Figure 5: (a) Subset of rows and columns from the "Criminology" table from UGen about Crime reports, and (b) values in the verified table. Each cell is color-coded based on its support score from the WikiTables data lake. Each cell also has the number of data lake tables that support that value (in red).

reproduced in the verified table (Figure 5b). However for Incidents 35 and 50 (second and third rows), the source table and Verified table have different values for "Victim Race". Neither of the "Victim Race" values in the source table have any support from the data lake, whereas the "Victim Race" values in the Verified Table have some support from the data lake. We find that all "Victim Race" values in the verified table have the same supporting table. Thus, "Black" and "White" are reasonable replacements for "Asian" under "Victim Race" in the source table. Note that some values in the verified table may not have any data lake support (e.g., "Victim Gender" values for Incidents 35 and 40). Gen-T aims to reproduce as many values as possible, and if Gen-T finds rows aligned at different columns (e.g., "Victim Age"), it may over-combine them and replace null values (supported by the data lake) to reproduce source table's values (e.g., Female gender). This creates a key-value pair that has no support from the data lake (e.g., Female Victim for Incident 40).

By comparing the degrees of support in source tables and Verified tables, we show that values in Verified Tables are supported by the data lake and can be used to replace values in the source table that have little or no support. Note that the corrections provided by our method do not take into account bias in the data or in the LLM [17, 24], although this would be an interesting direction for future research.

5 CONCLUSION AND FUTURE WORK

Table reclamation [10] has been proposed as a way of understanding how a set of tables may have been integrated. In this work, we propose a novel application of reclamation to the verification and correction of automatically generated tables, such as tables produced by generative AI. Our novel contributions include a method for computing the support of reclaimed data and a method for suggesting corrections to values that cannot be reclaimed. We have shown that our method outperforms an LLM-based data wrangling and correction approach [29]

Limitations: To find supporting tables from a data lake for a source table, we currently follow Gen-T [10] and rely on exact cell matches. However, this does not account for different syntactic representations of the same value. In addition, when we compute support values (Section 3.3), we simply find occurrences of source keys and key-value pairs. In future work, we need to develop a scoring function that considers how much support each row in the source table has from supporting tables, thus taking table context into account. Lastly, while we have preliminary case studies, we need to develop formal methods to evaluate table corrections and discuss table validation from a data lake.

Future Work: While this is a first step in validating LLM tabular outputs using a data lake and suggesting possible value replacements, there is great potential for future work. So far, methods for error correction have mainly relied on external sources such as Knowledge Bases or models [4, 8, 27, 29] or on LLMs themselves. Our work is a preliminary investigation into using data lakes for error correction. Our current support score gives a value a high score if it appears many times in a data lake. However, this does not mean that the value is accurate or trustworthy. Oftentimes, an error is propagated in a data science pipeline and thus appears in many tables. Future research requires studying the accuracy or trustworthiness of data (including understanding if tables are versions of each other) and aiming to reclaim them in a way that properly reflects these important properties.

ACKNOWLEDGMENTS

This work was supported in part by NSF under award numbers IIS-2107248, IIS-1956096, and IIS-2325632.

REFERENCES

[1] Patricia C. Arocena, Boris Glavic, Giansalvatore Mecca, Renée J. Miller, Paolo Papotti, and Donatello Santoro. 2015. Messing Up with BART: Error Generation for Evaluating Data-Cleaning Algorithms. *Proc. VLDB Endow.* 9, 2 (2015), 36–47.
[2] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2013. Methods for exploring and mining tables on Wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, Duen Horng Chau, Jilles Vreeken, Matthijs van Leeuwen, and Christos Faloutsos (Eds.). ACM, 18–26.
[3] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *ICDE*. 709–720.
[4] Bernardo Breve, Loredana Caruccio, Vincenzo Deufemia, Giuseppe Polese, et al. 2022. RENUVER: A Missing Value Imputation Algorithm based on Relaxed Functional Dependencies.. In *EDBT*. 1–52.
[5] Dan Brickley, Matthew Burgess, and Natasha F. Noy. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *WWW*. 1365–1375.
[6] Alexander Brinkmann, Anna Primpeli, and Christian Bizer. 2023. The Web Data Commons Schema. org Data Set Series. In *Companion Proceedings of the ACM Web Conference 2023*. 136–139.
[7] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. 2009. Data Integration for the Relational Web. *Proc. VLDB Endow.* 2, 1 (2009), 1090–1101.
[8] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*. 1247–1261.
[9] Vincenzo Cutrona, Jiaoyan Chen, Vasilis Efthymiou, Oktie Hassanzadeh, Ernesto Jiménez-Ruiz, Juan Sequeda, Kavitha Srinivas, Nora Abdelmageed, Madelon Hulsebos, Daniela Oliveira, and Catia Pesquita. 2021. Results of SemTab 2021. In *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching*, Vol. 3103. 1–12.

[10] Grace Fan, Roee Shraga, and Renée J. Miller. 2024. Gen-T: Table Reclamation on Data Lakes. In *ICDE*. 3532–3545.

[11] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *SIGMOD Companion*. 69–75.

[12] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. 2023. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *Proc. VLDB Endow.* 16, 7 (2023), 1726–1739.

[13] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *ICDE*. 1001–1012.

[14] Raul Castro Fernandez, Aaron J. Elmore, Michael J. Franklin, Sanjay Krishnan, and Chenhao Tan. 2023. How Large Language Models Will Disrupt Data Management. *Proc. VLDB Endow.* 16, 11 (2023), 3302–3309.

[15] Raul Castro Fernandez, Essam Mansour, Abdulhakim Ali Qahtan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018. Seeping Semantics: Linking Datasets Using Word Embeddings for Data Discovery. In *ICDE*. 989–1000.

[16] Sainyam Galhotra, Yue Gong, and Raul Castro Fernandez. 2023. Metam: Goal-Oriented Data Discovery. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 2780–2793.

[17] Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md. Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2023. Bias and Fairness in Large Language Models: A Survey. *CoRR* abs/2309.00770 (2023).

[18] Yue Gong, Zhiru Zhu, Sainyam Galhotra, and Raul Castro Fernandez. 2023. Ver: View Discovery in the Wild. In *ICDE*. 503–516.

[19] Xuming Hu, Shen Wang, Xiao Qin, Chuan Lei, Zhengyuan Shen, Christos Faloutsos, Asterios Katsifodimos, George Karypis, Lijie Wen, and Philip S. Yu. 2023. Automatic Table Union Search with Tabular Representation Learning. In *ACL*. 3786–3800.

[20] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2023. GitTables: A Large-Scale Corpus of Relational Tables. *Proc. ACM Manag. Data* 1, 1 (2023), 30:1–30:17.

[21] Ihab F Ilyas and Xu Chu. 2019. *Data cleaning*. Morgan & Claypool.

[22] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. arXiv:2401.04088 [cs.LG]

[23] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2023. SANTOS: Relationship-based Semantic Table Union Search. In *SIGMOD*.

[24] Hadas Kotek, Rikker Dockum, and David Q. Sun. 2023. Gender bias and stereotypes in Large Language Models. In *CI*. 12–24.

[25] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2023. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*. PMLR, 17564–17579.

[26] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc. VLDB Endow.* 3, 1 (2010), 1338–1347.

[27] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 1948–1961.

[28] Microsoft. 2024. https://www.microsoft.com/en-us/copilot, last accessed on Nov 21, 2023.

[29] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment* 16, 4 (2022), 738–746.

[30] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *Proc. VLDB Endow.* 11, 7 (2018), 813–825.

[31] OpenAI. 2024. Free Research Preview. https://chat.openai.com/, last accessed on Feb 25, 2024.

[32] Koyena Pal, Aamod Khatiwada, Roee Shraga, and Renée J. Miller. 2023. Generative Benchmark Creation for Table Union Search. *CoRR* abs/2308.03883 (2023).

[33] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Y. Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *SIGMOD*. 817–828.

[34] Roee Shraga and Renée J. Miller. 2023. Explaining Dataset Changes for Semantic Data Versioning with Explain-Da-V. *Proc. VLDB Endow.* 16, 6 (2023), 1587–1600.

[35] SlidesAI. 2024. https://www.slidesai.io/, last accessed on Nov 21, 2023.

[36] Nan Tang, Chenyu Yang, Ju Fan, Lei Cao, and Alon Halevy. 2024. VerifAI: Verified Generative AI. In *CIDR*.

[37] TPC. 2014. http://www.tpc.org/, last accessed on Nov 11, 2023.

[38] Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Canim. 2020. Web Table Retrieval using Multimodal Deep Learning. In *SIGIR*. 1399–1408.

[39] 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 *Advances in Neural Information Processing Systems 35*.

[40] Junwen Yang, Yeye He, and Surajit Chaudhuri. 2021. Auto-Pipeline: Synthesize Data Pipelines By-Target Using Reinforcement Learning and Search. *Proc. VLDB Endow.* 14, 11 (2021), 2563–2575.

[41] Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2023. Mixed-Type Tabular Data Synthesis with Score-based Diffusion in Latent Space. In *The Twelfth International Conference on Learning Representations*.

[42] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *SIGMOD*. 847–864.

[43] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *Proc. VLDB Endow.* 9, 12 (2016), 1185–1196.