ALT-GEN: Benchmarking Table Union Search using Large Language Models

Koyena Pal
Northeastern University
Boston, MA, USA
pal.k@northeastern.edu

Aamod Khatiwada
Northeastern University
Boston, MA, USA
khatiwada.a@northeastern.edu

Roee Shraga
Worcester Polytechnic Institute
Worcester, MA, USA
rshraga@wpi.edu

Renée J. Miller
Northeastern U. & Waterloo U.
Boston, MA, USA
miller@northeastern.edu

ABSTRACT

We consider the table union search problem which has emerged as an important data discovery problem in data lakes. Semantic problems like table union search cannot be benchmarked using only synthetic data. Our current methods for creating benchmarks for this problem involve the manual curation and human labeling of real data. These methods are not robust or scalable and perhaps more importantly, it is not clear how comprehensive the created benchmarks are. We propose to use generative AI models to create structured data benchmarks for table union search. We present a novel method for using generative models to create tables with specified properties. Using this method, we create a new benchmark containing pairs of tables that are both unionable and non-unionable, but related. We use this benchmark to provide new insights into the strengths and weaknesses of existing methods. We evaluate state-of-the-art table union search methods over both existing benchmarks and our new benchmarks. We also present and evaluate a new table search method based on large language models over all benchmarks. We show that the new benchmarks are more challenging for all methods than hand-curated benchmarks. We examine why this is the case and show that our new methodology for creating benchmarks permits more detailed analysis and comparison of methods. We discuss how our generation method (and benchmarks created using it) sheds more light into the successes and failures of table union search methods sparking new insights that can help advance the field. We also discuss how our benchmark generation methodology can be applied to other semantic problems including entity matching and related table search.

VLDB Workshop Reference Format:
Koyena Pal, Aamod Khatiwada, Roee Shraga, and Renée J. Miller. ALT-GEN: Benchmarking Table Union Search using Large Language Models. 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/gen.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

1 INTRODUCTION

David Paterson states that when a field has good benchmarks, we settle debates and the field makes rapid progress [33]. Traditionally, benchmark generation is done using synthetic data generators that precisely control parameters such as data size, data distributions, and correlations in data [17]. These parameters influence the standard DBMS performance metrics such as response time and through-put. However today, more and more data management challenges require not only the fast and scalable processing of data, but also an understanding of the semantics of data. A common example that we consider is *table union* – do two tables contain attributes and relationships with the same semantics so that they can be meaningfully unioned [29]. For this problem, the most important performance metric is accuracy. In comparing different methods, we would also like to understand their accuracy over tables with different data characteristics (for example, tables with more incompleteness or tables with longer textual attribute values).

The state-of-the-art in union search benchmarking is for researchers to find and curate (manually label) real datasets. Nargesian et al. [29] created the first labeled benchmarks (which we will call TUS-Small and TUS-Large in our work) using real open data (from government open data portals). These benchmarks have been reused for diverse applications, not just the table union search problem [4, 26, 27]. They took several large tables and sliced them horizontally and vertically to create tables that are unionable on some or all attributes. This approach was later extended by Khatiwada et al. [24] to create SANTOS-Small and SANTOS-LARGE benchmarks that were also used in other works [16, 19, 20, 25]. Table union search is not unique in its need for hand curated benchmarks. Another example is the T2K Gold Benchmark for evaluating matching systems, which was created from a large web table corpus [44] matched with properties from DBpedia and hand labeled. All these efforts require considerable human effort to find appropriate real data and to label the ground truth. Also, it is not clear how representative these benchmarks are. For example, while the TUS-Large benchmark [29] offers thousands of "labeled" tables, they originate from only 32 seed tables which are necessarily limited in the variety of semantics they capture (for example, the topics in

the tables). SANTOS-Large [24], which is large in size and based on real open data tables, offers only 80 human-labeled samples, not complete ground truth. Also, none of these benchmarks offer a set of non-unionable table pairs and assumes that *table pairs that are not labeled as unionable are not unionable*. This assumption is made for scaling, a human cannot label a quadratic number of table pairs for anything beyond a modest number of tables. But generative models give us an opportunity to address some of the limitations of hand labeled benchmarks.

Generative AI models have become increasingly popular especially in NLP, where models such as GPT3 [7] and ChatGPT [30] are used by millions of users on a daily basis. These models can be used as is (aka zero-shot learning [47]) or by providing a (small) set of examples that steer the model in the right direction with respect to the task at hand (aka in-context learning [7, 15]). *It is our thesis that these models could provide the key to making innovative new advancements in benchmarking semantic problems in structured data management.*

Contributions: We use generative AI models (which we will denote as LLMs) to create benchmark datasets for the table union task. Our contributions can be summarized as follows.

- **Automated LLM Table Generator (ALT-GEN) :** an automated framework to generate tables for table union search tasks based on a set of desired properties. Our framework includes a verification step that we use to study the extent and influence of hallucination on benchmark generation. In addition, we use a novel iterator to generate larger tables using LLMs which by definition have finite (limited) context.
- **New Table Union Search Benchmark:** we generate and share a new table union search benchmark UGEN-V2. UGEN-V2 contains 1050 tables on 50 topics ranging from World Geography to Veterinary Medicine, 1000 labeled table pairs (500 unionable/500 non-unionable) allowing fine-grained effectiveness analysis.[1]
- **Table Union Search Evaluation:** we evaluate and analyze existing table union search methods over the new and existing benchmarks. While the best search methods achieve a MAP of over 90% on existing benchmarks, we show that the new benchmark is more challenging. The best MAP values are around 70%, potentially triggering research into more robust methods.
- **A More Thorough Experimental Analysis:** previous work has used hand-labeled benchmarks as-is. We show that our ALT-GEN methodology lends itself to more robust benchmarking by considering other important parameters. For instance, we vary UGEN-V2 benchmark's sparsity (how many null values it contains) and evaluate methods on different table topics. We show that by using ALT-GEN, we can do ablation studies on these parameters to understand the table union search methods even better, something that has not been done before. This helps create a better understanding of the relative strengths and weaknesses of different search methods.
- **New Table Union Search Method:** in generating benchmarks using LLMs, an important question is whether an LLM is better at table union search (especially on its own benchmark) than the best existing methods. An LLM can easily be used to classify two tables as unionable or non-unionable, but not for table union search (given a query table, find tables within a massive repository that are unionable with it). So to answer this question, we present a new table union search method that uses a state-of-the-art search technique to find a set of candidates tables (Starmie [16]) and then uses an LLM to classify table pairs among the candidates as unionable/non-unionable. Our in depth analysis of the performance of this new method with existing methods highlights some important and interesting areas for future work and improvement.

In Section 2, we survey related work. Section 3 presents our methodology to generate benchmarks and the new benchmark. Section 4 provides analysis and evaluation.

2 RELATED WORK

We first present the state-of-the-art methods in table union search (including methods that appear in our experimental study). We then consider benchmark creation in data management tasks, including benchmark generation for semantic tasks like data cleaning and data integration. Finally, we introduce generative models.

Table Union Search: Given a query table by the user, table union search techniques find a set of data lake tables that can be unioned with the query table and potentially used to add new rows to the query table [29]. Nargesian et al. [29] considered two tables to be unionable if they a subset of their columns are unionable. Column unionability is determined by using an ensemble of three statistical tests based on value overlap, semantic overlap, and word embedding similarity of the column values. **D^{3}L** [4] extended that work [29] by considering, three additional attribute unionability measures (column header similarity, numerical value distribution, and regular expression similarity). **SANTOS** [24] considered the similarity of both columns and binary relationships between columns to make table unionability decisions. The binary relationships help SANTOS to understand table context better and omit the unioning of the tables having similar columns but different contexts [24]. **Starmie** [16] uses a contrastive-learning approach to capture the context of the entire query table rather than just the binary relationships. In very recent work, Hu et al. [20] used the contextualized representation of the relationships between the column pairs to capture the table contexts and use them to find unionable tables from the data lakes (however, open code is not available as of this writing). The benchmarks used (and created) in the respective papers and their limitations are discussed in the introduction. Importantly, their creation and evaluation involve a significant amount of manual annotation.

Benchmark Generation in Databases: Benchmark generation has been considered for many data management tasks. In data cleaning, Arocena et al. [2] introduced BART which can be used to add errors into clean databases and evaluate data-repairing algorithms. In data integration, iBench [1] generates schemas and schema constraints with arbitrarily large and complex mappings. These systems start with real data and metadata and systematically vary specific parameters that can influence the performance of cleaning or integration systems. In a similar spirit, we show how ALT-GEN can generate realistic data and use it to vary certain parameters (such as textuality) during the generation process. We also

[1] UGEN-V2 provides an important advancement over UGEN-V1, an early benchmark developed with ALT-GEN [32]. Both are used in our experiments, and both are described in Section 3.

consider how to post-process the generated result to vary other parameters such as sparsity.

Others have used knowledge graphs [13], Git repositories [21], and the web [5] to generate tabular benchmarks. For example, the SemTab challenge [13], which focuses on the semantic annotation of the tabular datasets using knowledge graphs, generates benchmarks using Wiki data for important intra-table tasks such as column type annotation and column-column relationship annotation. However, limited value coverage in the knowledge graph [24] could pose a challenge for creating diverse benchmarks and this is an area where generative models could be helpful. Another example is The Web Data Commons project [5] which extracts schema.org [18] data from the Web using Common Crawl [12]. This project yielded several successful benchmarks for multiple tasks such as Product Matching [34]. The extraction is limited to a specific schema and still requires manual annotation.

A recent data discovery benchmark includes Lakebench which contains 8 open datasets for joinability and unionability tasks [39]. Different from our work where we generate tables for *table search* task using a generative model, they manually label existing open datasets for fine tuning large language models. For some benchmarks, they also use entities and classes in Knowledge graphs to create and annotate tables.

Generative Models and LLMs: Generative models generate new data instances making them appealing for benchmark generation. They have been used to augment training data in various related tasks such as commonsense reasoning [48], event detection [43], text classification and summarization [10]. Recently, researchers have coined the term large language models (LLMs) which refers to generative language models in general. We use Mistral AI's Mixtral-8x7B-Instruct [22] for UGEN-V2 and Open AI's GPT3 [7] for UGEN-V1 benchmark generation and also experiment with GPT2-xl [37], Alpaca [40] and Vicuna [50] (open source via Hugging Face [46]). LLMs have been used within the data management community extensively. For example, Arora et al. [3] use LLMs to generate structured views of semi-structure data lakes. Others use LLMs to extract knowledge graphs [11, 45]. Trummer used GPT-3 to generate code for query processing [41, 42]. Recently, some attempts were made to solve other data management tasks such as information extraction [6] and entity matching [34] using prompting and in-context learning. But to the best of our knowledge, generative models have not be used in generating benchmarks for semantic data management tasks.

Note that our preliminary work is available on ArXiV [32]. In the current work, we standardize the benefits and challenges of generative benchmarks and present a framework to create larger and verified tables.

3 GENERATIVE UNION BENCHMARKS

We now present the opportunities that LLMs present for Benchmark Generation as well as some of the important challenges that must be overcome to make them effective.

3.1 The Promise

Generative models are designed to generate realistic data. This promise means they may be able to generate (potentially) better semantic database benchmarks than humans are able to create, label, and curate themselves and in less time.

Realistic and Diverse Data: Benchmarks for semantic data management problems should in general contain *realistic* and semantically meaningful (not necessarily *real*) tuples and structured tables. All union benchmarks to date have been created from real open data sets rather than using synthetic data, though this data is often manipulated (for example by chopping up tables and tuples into smaller pieces randomly). Hence, the benchmarks contain realistic data though due to the manipulation, individual facts (tuples) may not be verifiable or *real* in that they appear in a known knowledge base. LLMs are good candidates for benchmark generation as they can generate realistic data. We do not require that this data be verified as being accurate. We use the temperature parameter to ensure the model is set to be creative to generate a diverse set of tables, on a more diverse set of topics than has been feasible using manual benchmark creation.

Balancing the Hardness of Unionable and Non-unionable Examples: In addition, LLMs provide a new opportunity for union search benchmarking that has not been explored before. The state-of-the-art in benchmarking is to select a set of disparate large tables, called originating tables, and chop them up. The created tables that originate from the same table are the (labeled) unionable pairs. All others are assumed to be non-unionable implicitly. To make this assumption realistic, each of the originating tables is choosen to be from a different topic so that it is unlikely that any subset of it would be unionable with a subset from a different originating table. Consider what this means for TUS-Small, a benchmark containing 1530 tables of which just over 200K pairs are labeled as unionable. The remaining 2 Million pairs are (implicitly) assumed to be non-unionable. Of course this has never been verified by hand. So to ensure this is the case, the original source tables must be very different indeed, making the non-unionable cases perhaps too easy. In particular, there are no non-unionable table pairs on the same topic. This is a gap in our community's benchmarks that we show can be overcome by LLMs.

Varying Data Characteristics: Finally, an opportunity provided by LLMs is the ability to vary additional data characteristics such as textuality (the average length of string values), the ratio of numeric to non-numeric attributes, and the incompleteness or sparsity (number of nulls) of tables. These are characteristics that can influence the accuracy of different methods. In our benchmark generation methodology (Section 3), we show how these characteristics can be added to the prompts. However, due to space limitations, we do not vary these parameters in the experiments, rather we report their values for our new benchmark and existing benchmarks for comparison. This report is motivated by Primpeli and Bizer [36], who presented a set of similar properties that should be reported within (hand-curated) entity-matching benchmarks (they also do not vary these characteristics systematically). To explore the potential for varying such characteristics, we perform an ablation study that injects different amounts of nulls (degrees of sparsity) into both positive and negative example table pairs (Section 4.3.1).

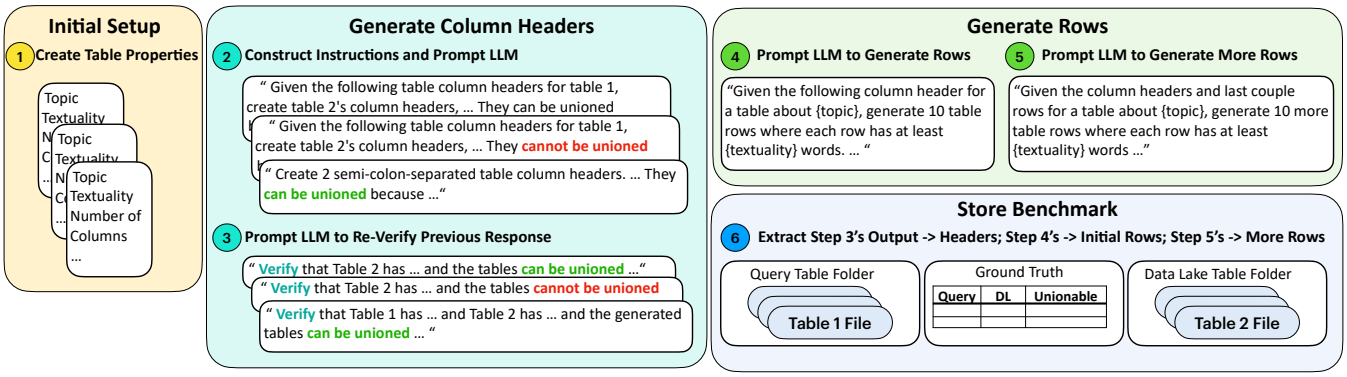


Figure 1: Pipeline for generating benchmarks using Large Language Models

3.2 The Challenges

Despite these opportunities, there are important challenges in the use of LLMs for benchmarking semantic database problems.

Generating Large Data: LLMs have a limited context length, which includes the number of tokens they can read and generate in a single run. This means it is challenging to generate large tables. To address this issue, in our framework, ALT-GEN, we break down our table generation into three steps. We first generate the table's column headers. Second, we include the table's column headers and instruct the model to generate T_{rows} rows where this parameter is set based on the size of the output the LLM can generate. In the third and following prompts, we sample and prompt the previously generated rows and instruct the model to generate more rows.

Model Hallucination: One of the well-known issues in LLMs is that they do not always follow the prompt and so may produce data that is not accurate. This phenomenon is generally termed model hallucination. We show how we can address this by adding a verification step, where we include an additional prompt for the model to double-check its response and verify whether it generated a (non-)unionable pair of as requested. We present an empirically analysis of our verification method in Section 3.4 comparing a benchmark created without verification to one that uses verification.

Reducing Prompt Sensitivity: An LLM's response to an input prompt can change with even subtle changes to the prompt [23, 28, 49]. Hence, it can be challenging to come up with robust prompts. ALT-GEN addresses this using systematic offline trials with prompt instruction text variations.

Algorithm 1: Benchmark Generation

Input: $\mathcal{S} = \{S_1, S_2, \ldots S_m\}$, a list of topics
//Language Model Initialization
$\mathcal{LM} \leftarrow$ Model(temp=0.7, repPenalty=2.0, doSample=True, ...)
$Q_{COLS}, DL_{COLS} \leftarrow$ `GenerateColumnHeaders`$(\mathcal{S}, \mathcal{LM})$
$\mathcal{D} \leftarrow [T_1, ...T_k]$ where $T \in Q_{COLS}, DL_{COLS}$
$D_{Rows} \leftarrow$ `GenerateFirstNRows`$(\mathcal{D}, \mathcal{LM})$
`AddMoreRows`$(D, D_{Rows}, \mathcal{LM})$

3.3 Unionability Benchmark Generation

Figure 1 illustrates ALT-GEN's pipeline for generating tables using LLMs. Through a series of phases, we prompt an LLM to generate pairs of tables – first their column headers, and then their rows with the following set of features customized for each table: topic, shape, and the textuality rate. In Algorithm 1, we showcase the algorithmic breakdown of the same. To generate tables for a unionability benchmark, we add additional properties to the prompt such as whether the table pairs are supposed to be unionable or not.

We now provide an example illustrating prompts and the corresponding generated tables.

Example 1. *Figure 2 provides an instance of how table-pairs can be generated. Starting off with generating a unionable-table pair, we generate both query table and unionable table column headers. Using the same query table headers, we generate non-unionable table column headers. Once all the table headers are generated, we generate each table's first set of rows. The top-most table on the right side of the figure is the generated query table. The bottom-most left side of the figure is the generated unionable table, and beside that table, is the resulting non-unionable table. The generated tables contain realistic data. For example,* `White House`*'s architect is* `James Hoban`*. While this data point is historically accurate, in the benchmark, there can be instances were this is not the case. For the union search problem, this is not an issue (and we do not need to verify the current accuracy of facts in tables). What is important is that the data is realistic and about the given topic. In some real-world applications where real data can be considered as a privacy leakage, this may be an advantage [8, 9].*

The unionable table pairs (bottom-left and top-right) are related to `Architecture` *and share (at least) 6 unionable columns, out of which* `Building Description`, `Style`, `Build Year`, *and* `Designer` *are visible in the figure. Notably, the column values overlap (e.g.,* `Sydney Opera House`*), but not completely (for instance, subset of the year values match) and contain multiple data types (e.g.,* `Building Type` *is a string and* `Year Built` *is a date).*

The non-unionable table pairs (right) are also related to `Architecture` *and do not share any unionable columns. Note that current methods for unionability consider tables unionable even if they share a single unionable column [29], but of course the prompt could be changed when testing other unionability solutions. We envision such examples to be the most challenging ones for contemporary methods which are built on top of identifying semantics, which can*

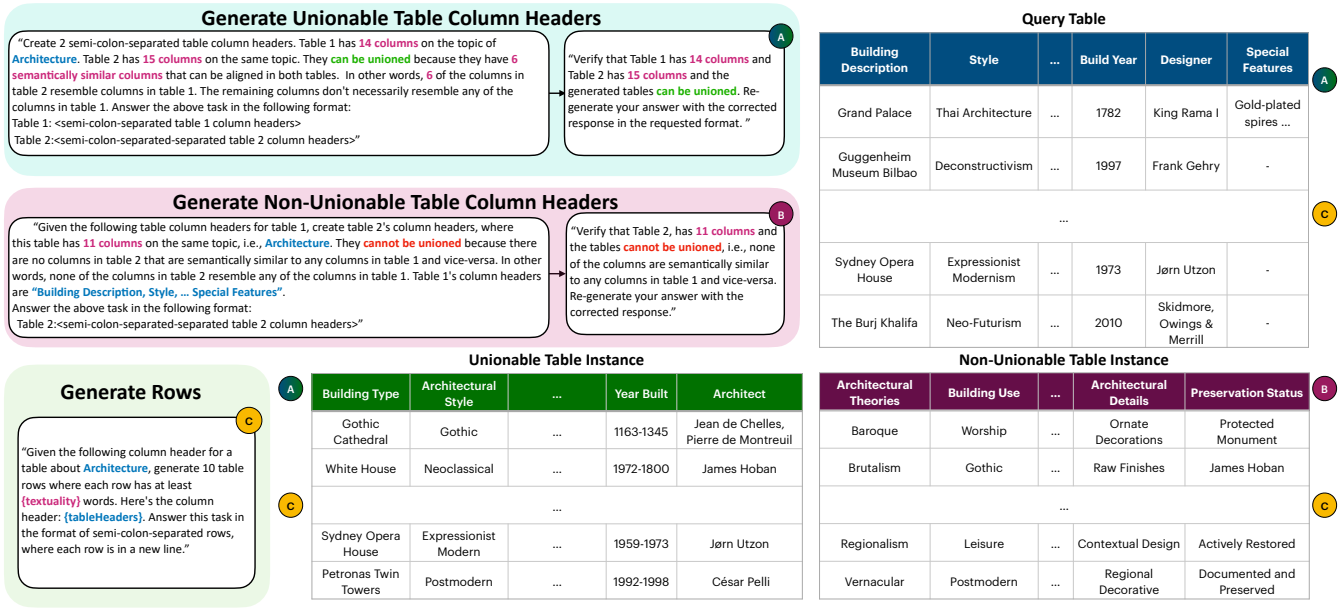


Figure 2: Example of generating unionable and non-unionable table pairs using ALT-GEN. The A, B, and C indicators show where the prompt's outputs are used in the creation of the tables.

be mistakenly thought of as topic. Also, here we see some interestingly missing values (in `Special Features`*) –another property we aimed for in a benchmark (incompleteness).*

3.4 New Generative Benchmarks

In an initial investigation, we generated **UGEN-V1** [32] using GPT3 as the LLM. We generated 1000 pairs of tables (half unionable, half non-unionable), covering 50 topics ranging from World Geography and Art History to Genealogy and Veterinary Medicine. For each topic, we generated 1 query table and 20 data lake tables (10 unionable, 10 non-unionable with the query table). In UGEN-V1, we did not attempt to create larger tables. Rather, we limited the number of rows to eight and generated eight rows from a single prompt to avoid scalability issues. We also did not verify the generated data.

UGEN-V2 was created using ALT-GEN and Mixtral-8x7B-Instruct (as the LLM model). Similar to UGEN-V1, we generated 1000 pairs of tables (half unionable, half non-unionable). This took approximately 10 hours to generating column headers and 24 hours for generating up to first 10 rows for the 1050 tables (50 query tables on 50 different topics and 1000 data lake tables). An additional 24 hours were used to generate additional rows (around 100 rows) for a sample of tables. We ran these prompt requests on a shared cluster of 8 A100 GPUs. We also created 5 versions of each table with different levels of sparsity (0%, 5%, 10%, 15% and 20% of the values in a table are replaced by nulls). The benchmark and our generation code for ALT-GEN are available.[2]

Benchmark Validation: This is the first use of LLMs for table union search benchmark generation. Thus, in this work, we manually validated both UGEN-V1 and UGEN-V2 benchmarks. Recall UGEN-V1 does not use our verification step, while UGEN-V2 does. For a total of 1000 evaluation cases (50 query tables from 50 different topics in the benchmark, each having 10 unionable and 10 non-unionable data lake tables labeled in the ground truth), UGEN-V2 achieves an overall accuracy of 97%. Within that, the accuracy with respect to unionable (query and data lake) table pairs is 98% and that with respect to non-unionable table pairs is 96%. UGEN-V1, on the other hand, achieves 77% overall accuracy, with 90% for unionable table-pairs, and 64% for non-unionable tables.

To understand the performance even better, we evaluated the accuracy for all 50 topics in UGEN-V2 separately. Out of them, 49 (respectively, 38) topics achieve 90% or higher accuracy only considering unionable pairs (respecively, non-unionable pairs). This shows that it is more challenging to create the non-unionable table pairs than the unionable ones. For comparison, we manually evaluate the ground truth accuracy of the existing TUS-Small benchmark. As the unionable columns of the unionable tables in the TUS benchmark are generated from the same seed table [29] and hence, are always accurate, we evaluate the accuracy of non-unionable table pairs. We randomly sample 100 non-unionable table pairs and evaluate them. Overall, the accuracy of non-unionable pairs in the TUS Benchmark is 62%. This shows that our new automated approach generates benchmarks with significantly better accuracy without manual work in a more challenging setup of generating non-unionable table pairs from the same topic.

3.5 Comparison of Unionability Benchmarks

Table 1 overviews the new benchmarks (UGEN-V1, UGEN-V2) as well as other unionability benchmarks. We provide the following features of each benchmark:

 # Tables: Total number of benchmark tables.

 # Average Shape: Average number of rows and attributes in the tables, for query tables and also for data lake tables.

[2] https://github.com/northeastern-datalab/gen

Table 1: Overview of Union Table Search Benchmarks

Benchmark		SANTOS-Small	SANTOS-Large	TUS-Small	TUS-Large	UGEN-V1	UGEN-V2
# Tables		550	11166	1530	5044	1050	1050
# Average Shape	**Query**	21402 x 12	12917 x 13	4423 x 13	1996 x 12	8 x 11	107 x 13
`rows x columns`	**Datalake**	6921 x 11	7685 x 11	4457 x 10	1915 x 11	8 x 10	19 x 13
Size	**Query**	129 MB	143 MB	139 MB	2 GB	205 KB	2 MB
	Datalake	442 MB	12 GB	1 GB	2 GB	4 MB	8 MB
# Labeled Pair	**Unionable**	695	80	230921	1202826	500	500
	Non-unionable	0	0	0	0	500	500
Attributes `(query,datalake)`	**% Short Str**	(13%, 11%)	(8%, 7%)	(10%,25%)	(31%, 35%)	(1%, 2%)	(2%, 1%)
	% Medium Str	(14%, 16%)	(19%, 21%)	(10%,12%)	(20%, 19%)	(11%, 9%)	(10%, 8%)
	% Long Str	(49%, 48%)	(48%, 45%)	(69%, 53%)	(41%, 39%)	(81%, 82%)	(75%, 74%)
	% Num	(22%, 23%)	(16%, 18%)	(11%,9%)	(6%, 6%)	(6%, 4%)	(11%, 13%)
	Avg. Density	(95%, 93%)	(85%, 76%)	(94%,96%)	(87%, 89%)	(94%, 94%)	(96%, 94%)
	% Small Domains	(64%, 58%)	(65%, 50%)	(81%, 87%)	(82%, 83%)	(5%, 7%)	(19%, 8%)
	% Medium Domains	(13%, 15%)	(15%, 18%)	(6%, 5%)	(10%, 9%)	(11%, 10%)	(26%, 14%)
	% Large Domains	(23%, 27%)	(20%, 32%)	(13%, 8%)	(8%, 8%)	(84%, 84%)	(54%, 78%)

Size: Total size/storage space of the benchmark.
 # Labeled Pairs: For each benchmark, there is a ground truth file that indicates which pairs of tables are unionable or not. For existing benchmarks, this can be a non-exhaustive list since non-unionability, for instance, can be inferred to hold for all pairs that are not stated to be unionable (though this may include false negatives e.g., if tables happen to share one or more address or date attributes). We report the number of unionable and non-unionable pairs explicitly labeled in the ground truth.
Attributes: In the attributes section of the table, we give the following properties of the query tables (first number) and data lake tables (second number). All are averages over the total number of attributes except density which is an average over the total number of cell values in the tables.

(1) % Short Str: Percentage of string attributes that have an average length less than 3.
(2) % Medium Str: Percentage of string attributes that have an average length less than 6, but ≥ 3.
(3) % Long Str: Percentage of string attributes that have an average length greater than or equal to 6.
(4) % Num: Percentage of numerical type attributes.
(5) Avg Density: Avg percentage of non-null values present in attributes.
(6) % Small Domains: Percentage of attributes where the number of unique values is less that 20% of the table size.
(7) % Medium Domains:Percentage of attributes where the number of unique values is between 20% and 50%.
(8) % Large Domains: Percentage of attributes where the number of unique values is more than 50%.

Compared to past benchmarks, our two new benchmarks have relatively smaller tables in terms of average number of rows and attributes in both query and data lake sets of tables although ALT-GEN permits further scaling of both. However, even with this modest size, they are more challenging benchmarks as we will show in the next section. Unlike previous benchmarks, our benchmarks have higher textuality rate, which is evident in the comparatively higher long string ratio, and higher unique domain values (the hand-labeled benchmarks are created from 10 seed tables, which means many domain values are reused). These are contributing factors that may make our benchmarks more difficult.

4 ANALYSIS AND EVALUATION

We now perform empirical analysis over existing benchmarks and the new generative benchmarks.

4.1 Experimental Setup

We run experiments using Python 3.8 on a server with A100 80GB. We host the large language models locally in the aforementioned GPU. For the models unavailable to host locally, we use their open APIs. Our experiments aim to answer specific questions.

- How do the current union search methods perform on the new UGEN-V2 benchmark?
- Can generative models understand unionability?
- How is the performance of union search techniques impacted by tables sparsity (incompleteness)?
- How does the performance of union search techniques differ on specific topics?

We now detail the tasks, benchmarks, methods and evaluation metrics.

4.1.1 Tasks. We consider two tasks. The first is the traditional search problem for which there are numerous solutions in the literature [4, 16, 24, 29].

Table Union Search. Given a query table, Q, and a set of data lake tables, $\mathcal{T} = \{t_1, t_2, ...t_m\}$, find the top-$k$ data lake tables in $\mathcal{T}$ that are most unionable with Q.

Table Union Classification. Given a pair of tables T_1 and T_2, determine if they are unionable or not. This second problem is motivated by our use of LLMs to generate a unionability benchmark. Since we are claiming LLMs can produce unionable tables and

benchmark table union search solutions, can we also use an LLM to classify whether two tables we give to it are unionable or not? For example, we can use labeled pair of tables from our own UGEN-V1 benchmark or hand-labeled pairs from any of the existing benchmarks. This can be viewed as a sanity check on whether the notion of unionability used by an LLM is a reasonable one conforming to definitions used in current research.

4.1.2 Union Search Methods and Benchmarks. We evaluate the publicly available recent union search methods over the existing and new benchmarks.

D^{3}L [4]. Bogatu et al. extended TUS [29] to use not only word embeddings, knowledge graph mappings, or value overlap, but also column header similarity, distributions for numerical columns, and regular expressions.

We used D^3L's publicly available code.[3] For a fair comparison, we do not use the column header similarity metric since the existing benchmarks use identical schema names for unionable columns [24, 29].

SANTOS [24]. SANTOS uses column semantics and the semantics of relationships between column pairs to search for the unionable tables. SANTOS only uses column values and does not use metadata like column headers. To find column semantics and relationships semantics, SANTOS uses an external knowledge base and a synthesized knowledge base created using the data lake itself. To run SANTOS, we use the public code provided with the paper.[4]

Starmie [16]. Starmie is a recent self-supervised table union search technique based on contrastive learning. Starmie captures the table context in the form of contextualized column embeddings and uses them to perform table union search. We reproduced Starmie following the instruction in its open implementation.[5]

Starmie-LLM. LLMs cannot be used directly for the search task but, as mentioned, can be used for union classification. Therefore, to assess LLMs in the table union search task, we use an existing table union search method to search for a set of candidate unionable tables for each query table. Then we prompt an LLM to classify whether the query table is unionable with each of these candidate tables. This two-phase approach is very common for information retrieval applications [14, 38] in which two models are applied consecutively. In our experimental setup, we use Starmie to search for a larger number of candidate unionable tables. Then we prompt an LLM to classify each query-candidate table pair as unionable or not by asking the following question:

```
Are the following tables unionable? Answer in the
following format: Unionable: {yes/no}
```

We use recent LLMs that have shown promising performance in other generative tasks. Specifically, we use GPT2-XL[6], Alpaca (7 Billion parameters)[7], and Vicuna (7 billion parameters)[8] in our experiments. We denote respective LLM variations using Starmie-GPT2-XL, Starmie-Alpaca, and Starmie-Vicuna.

[3] https://github.com/alex-bogatu/d3l
[4] https://github.com/northeastern-datalab/santos
[5] https://github.com/megagonlabs/starmie
[6] https://huggingface.co/gpt2-xl
[7] https://huggingface.co/circulus/alpaca-7b
[8] https://huggingface.co/lmsys/vicuna-7b-v1.3

Table 2: P@k, MAP@k and R@k of table union search methods over different benchmarks.

Benchmark	Method	*MAP@k*	*P@k*	*R@k*
TUS-Small k=60	D^{3}L	0.79	0.77	0.21
	SANTOS	0.88	0.81	0.23
	Starmie	**0.94**	**0.82**	**0.27**
	Starmie-Vicuna$_{\text{Zero}}$	0.89	0.70	0.24
	Starmie-Vicuna$_{\text{Optim}}$	**0.94**	0.80	**0.27**
SANTOS-Small k=10	D^{3}L	0.52	0.58	0.42
	SANTOS	0.94	**0.91**	**0.69**
	Starmie	**0.95**	**0.91**	0.68
	Starmie-Vicuna$_{\text{Zero}}$	0.81	0.68	0.50
	Starmie-Vicuna$_{\text{Optim}}$	**0.95**	0.90	0.67
UGEN-V1 k=10	D^{3}L	0.26	0.19	0.19
	SANTOS	0.56	0.46	0.46
	Starmie	**0.61**	**0.51**	**0.51**
	Starmie-Vicuna$_{\text{Zero}}$	0.44	0.32	0.32
	Starmie-Vicuna$_{\text{Optim}}$	0.57	0.48	0.48
UGEN-V2 k=10	D^{3}L	0.15	0.13	0.13
	SANTOS	0.43	0.27	0.27
	Starmie	**0.71**	**0.56**	**0.56**
	Starmie-Vicuna$_{\text{Zero}}$	0.65	0.48	0.48
	Starmie-Vicuna$_{\text{Optim}}$	**0.71**	**0.56**	**0.56**

LLMs can function either as they are (zero-shot) or can be directed towards specific tasks by providing a few examples (in-context learning). For comparison with other methods, we denote zero-shot versions denoted as Starmie-LLM$_{\text{Zero}}$ and in-context versions with an optimal number of examples (meaning the best performance against other in-context versions) as Starmie-LLM$_{\text{Optim}}$.

4.1.3 Evaluation measures. Following the literature [4, 24, 29], we use Precision@k (P@k), Recall@k (R@k) and Mean Average Precision (MAP@k) to evaluate the effectiveness of table union search techniques. Consistent with prior work [16, 24], we run experiments with k less than the ground truth size to ensure that there are enough true results available in the data lake when searching for top-k unionable tables per query table. In such cases, a Recall@k of 1 is not possible since all unionable tables cannot be returned. We refer to the best possible Recall@k as the IDEAL. Furthermore, unlike existing benchmarks, our new benchmarks also contain labeled non-unionable pairs. So, we also measure Accuracy (ACC) and Corner Case Ratio (CCR) in these benchmarks. Let, TP, FP, TN, and FN represent the True Positives, False Positives, True Negatives, and False Negatives returned by a method over a benchmark. Then,

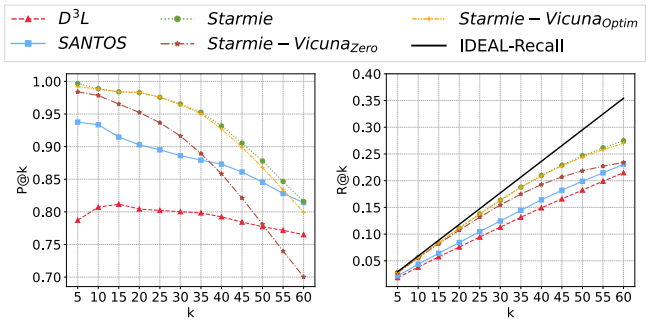
$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \qquad CCR = 1 - ACC \qquad (1)$$

We further use a confusion matrix to illustrate the specific TP, FP, TN, and FN values.

4.2 Union Search Effectiveness

Now we evaluate the performance of various table union search methods using both existing and new benchmarks. Specifically, we compare D^{3}L, SANTOS, Starmie, and Starmie-LLMs. Among

Starmie-LLMs, we select the best-performing Starmie-Vicuna's zero-shot version (Starmie-Vicuna$_{Zero}$) and optimal-shot version (Starmie-Vicuna$_{Optim}$) as it shows a balanced performance in both versions. In the ablation study (Section 4.3), we discuss the results of other Starmie-LLM variations and their different versions. The evaluation metrics used are MAP@k, P@k, and R@k. Following previous work [24, 29], the maximum value of k is chosen for each benchmark, based on the number of unionable tables available in the data lake for query tables. For TUS-small, we select k up to 60. For SANTOS-small, we go up to k = 10. Accordingly, in Starmie-LLM, we select 70 candidates for the TUS-Small and 20 candidates for SANTOS-Small, UGEN-V1 and UGEN-V2. The effectiveness results for the maximum value of k on each benchmark is presented in Table 2. We **bold** the score of the best-performing method on each measure and benchmark. Further results for other smaller values of k are plotted in Fig. 3 with different values of k in horizontal axes, the evaluation metrics in vertical axes, and the methods encoded using different colors and line styles. As noted, the recall cannot be perfect if k is smaller than the ground truth size [24]. So, we show the IDEAL-RECALL line that indicates the maximum possible recall for each value of k.



(a) Average $P@k$ on TUS-Small (b) Average $R@k$ on TUS-Small

(c) Average $P@k$ on Santos-Small (d) Average $R@k$ on Santos-Small

(e) Average $P@k$ on UGEN-V2 (f) Average $R@k$ on UGEN-V2

Figure 3: Effectiveness of baselines in different benchmarks

Curated Benchmarks. The performances of all methods follow a similar trend in both TUS-Small and SANTOS-small benchmarks. Specifically, Starmie mostly stands out in terms of MAP@k, P@k, and R@k followed by Starmie-Vicuna$_{Optim}$, Starmie-Vicuna$_{Zero}$, SANTOS and D^{3}L. As Starmie captures the entire table context using contrastive learning, it seems to understand the table semantics better in both benchmarks. Furthermore, when we steer Starmie-Vicuna to understand unionability by providing an optimal number of in-context examples, its MAP increases by 5% over the zero-shot version in the TUS-small benchmark, and by 14% in the SANTOS-small benchmark matching the performance of the best-performing Starmie. This indicates that the out-of-the-box LLMs are not good enough to understand unionability, but they can be taught to do so by using in-context learning. Moreover, SANTOS also shows a comparable performance against Starmie and Starmie-Vicuna$_{Optim}$ in both benchmarks and is the best method in SANTOS-small in terms of P@10 and R@10.

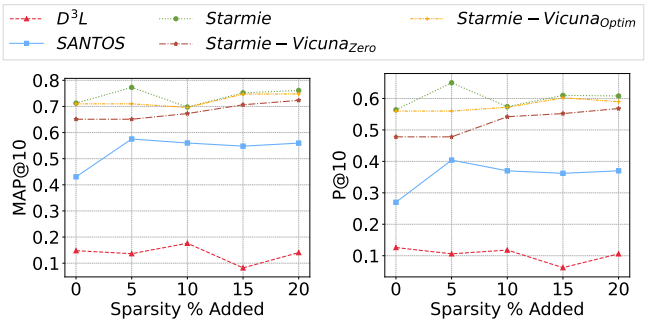
Generated Benchmarks. On the new UGEN-V1 and UGEN-V2 benchmarks, Starmie continues to outperform all other methods across all three evaluation metrics, followed by Starmie-Vicuna$_{Optim}$, SANTOS, Starmie-Vicuna$_{Zero}$, and D^{3}L. However, it is important to note that the performance of all methods shows a significant drop when compared to their performance on the SANTOS-Small and TUS-Small benchmarks. For instance, Starmie's MAP@k on UGEN-V2 for all values of k drops by over 20% compared to its MAP@k on both the SANTOS-Small and TUS-Small benchmarks.

Finally, the lower performance achieved by Starmie-Vicuna$_{Zero}$, the best among the zero-shot Starmie-LLM variations, against the classical table union search methods, indicates that even advanced LLMs like Vicuna face challenges in the new benchmark. This shows ample opportunity for creating better table union search methods.

4.3 Ablation Study

Now we perform fine-grained analyses over the new benchmark to get better insights into the impact of different factors in the table unionability search. We also study performance on both unionable and non-unionable pairs to see if the (perhaps harder) labeled non-unionable pairs may account for this drop in performance on the new UGEN-V2 benchmark. The trends are similar in the UGEN-V1 benchmark and they are available in our repository.[2]

4.3.1 Sparsity. One of the benchmark properties that we care about is sparsity, which relates to the number of null or missing attribute values in the tables. Within the ALT-GEN framework, this property is controlled by a script that randomly removes values in a table until the desired sparsity is reached. Fig. 4 illustrates how different methods can handle benchmarks with different rates of sparsity. We vary sparsity from 0% to 20% on the X-axis and report MAP@10 and P@10 in y-axis. SANTOS, Starmie, and Starmie-Vicuna are not impacted by sparsity. However, the performance of the column-based approach (D^{3}L) goes down from the 0%-sparsed version when we increase sparsity. For example, its MAP@10 reduces by 50% when we increase sparsity to 15%. This gives us an interesting insight that the methods which make unionability decisions by capturing table context rather than just considering individual columns independently are more tolerant towards sparsity.



(a) Average *MAP*@10 **(b) Average** *P*@10

Figure 4: Sparsity variations of UGEN-V2

4.3.2 Topic-Based Analysis. One of the benefits of the ALT-GEN framework is that we can generate unionable and non-unionable table pairs based on topics. This enables us to perform a topic-based analysis and understand how each baseline performs on various topics. For a comprehensive analysis, we also report MAP@10 by each union search method on all 50 topics in Figure 5. In the repository [2], we also highlight the top-5 and bottom-5 topics out of 50 topics based on MAP@10 for each method.

Except for Starmie and its LLM variation (Starmie-Vicuna), it is interesting to see that there is not much overlap in the top-5 best topics of each method. Even more interesting, there is an overlap between the top-5 topics of one method and the bottom-5 topics of another method. For example, `Sports` is the second best topic `Starmie` and `Starmie-Vicuna`, but the second worst topic for D^{3}L and third worst topic for `SANTOS`. This signifies that each method captures different properties in the table to infer table unionability and an easy topic for one method could be difficult for another method. The ALT-GEN framework allows users to delve into the characteristics of tables in different topics to understand such differences. In this case, the shorter strings mostly representing entities (`Artwork, Artist`) in `Art-History` may give a knowledge-graph approach like SANTOS an advantage while the longer descriptive strings in `Economics` (`policy details`) may give D^{3}L or TUS that use word embedding vectors an advantage.

This result suggests that future work on building domain-specific search methods could be useful along with combining the strength of multiple methods to build a single highly-effective table union search method.

Additional interesting insights are visible in Figure 5 which allows a comparison among methods for a given topic. For example, `Religion` achieves a fair MAP@10 score among all methods except SANTOS which achieves a score of less than 0.5. This can be attributed to poor coverage of their knowledge base in this topic, which is key for a successful utilization of SANTOS [24]. Another apparent example is the topic of `Language` for which D^{3}L fails to find a single unionable table (MAP@10=0) while others were very successful with significantly higher MAP@10 scores. Finally, another interesting insight is the topic-specific benefit of in-context learning, comparing Starmie-Vicuna$_{Zero}$ (Figure 5 (d)) Starmie-Vicuna$_{Optim}$ (Figure 5 (e)). For example, along side topics such as `Psychology` and `Literature` which are unaffected, topics such as `Environment` and `Astronomy` are significantly improved with in-context learning and, interestingly, the performance over topics such as `Gardening` and `Business` declines.

4.3.3 Non-Unionable Pair Analysis. ALT-GEN has the ability to generate labeled non-unionable table pairs from the same topic, something that was previously overlooked in unionability benchmarks [24, 29]. As reported in Table 1, we have 50 query tables and each query has 10 unionable data lake tables and 10 non-unionable data lake tables, all of which are on the same topic and labeled in the ground truth. The other tables that are unlabeled with respect to the query are non-unionable as they are semantically different, not even from the same topic. In this section, we analyze the potential impact of non-unionable table pairs from the same topic on the performance of the union search methods by creating confusion matrices using the 1000 labeled unionable and non-unionable pairs and reporting accuracy over them vs. overall non-unionable tables.

The availability of "non-unionable" table pairs from the ground truth, allows us to also extract true negatives in addition to the traditional true positives. Thus, we generate confusion matrices (provided in a technical report[9] for space considerations). By analyzing these matrices, we find that D^3L seems to be the best at detecting non-unionable tables (predicts 467 out of 500 non-unionable pairs accurately). However, if we look at true positives, D^{3}L has the least number, around two times fewer than the second-least performing SANTOS. This means that the false positives in D^{3}L's results are other unlabeled non-unionable tables from random topics rather than the labeled "non-unionable" tables from the same topic. Starmie and Starmie's LLM variations have high true positives and relatively lower true negatives. This means that their false positives are mostly non-unionable tables from the same topic, as we discussed in Example 1. This study does reveal that by labeling non-unionable pairs in UGEN-V2, we have been able to create difficult cases that lead to false negatives for all methods. We also get a very interesting insight into table unionability – it is important that our benchmarks are not just testing if a method can find tables on the same topic. Instead, the search methods should also separate non-unionability among the same topic tables.

4.4 Discussion and Future Work

In ALT-GEN, we use the massive knowledge present in LLMs to generate table pairs. Of course, there remain certain limitations posed by this framework. In ALT-GEN, it is not guaranteed that the size and type of table pairs are exactly as requested in a prompt. For instance, it is possible that an LLM generates a table with a different number of rows than requested in the prompt.

As future work, including examples for each table-pair generation could be more helpful to create more unionability-specific tasks. While this can be limited due to token size limits in LLM prompting, there are many new LLMs such as GPT4 [31], that have very recently been released and have higher token limit tolerance. In addition, as ALT-GEN has showcased, it is possible to breakdown queries into smaller sub-queries to ultimately generate a desired benchmark. Apart from this direction, the examples showcase the ability for LLMs to generate other inter-table tasks such as joinability or related table search, or entity matching. While LLMs are being

[9]The technical report can be found at the GitHub link: https://github.com/northeastern-datalab/gen

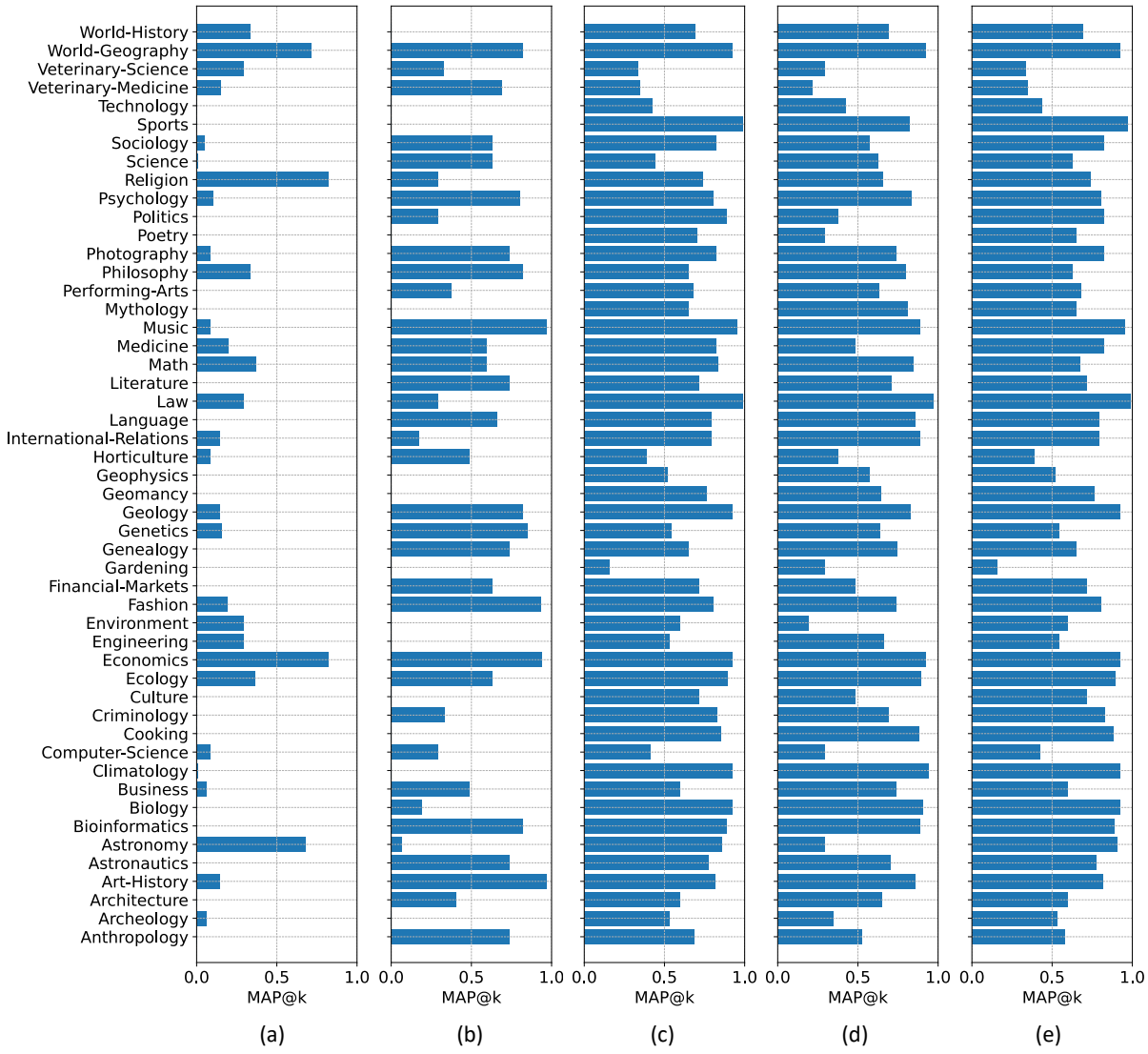


Figure 5: Topic-based MAP@k (k=10) by (a) D^{3}L (b) SANTOS (c) Starmie (d) Starmie-Vicuna$_{Zero}$ (e) Starmie-Vicuna$_{Optim}$ methods in UGEN-V2 benchmark

used to solve some of these tasks [35], we believe there is great potential in also generating benchmarks for them. Lastly, using LLMs as new table union search method relies on current methods such as Starmie to provide a small set of candidate unionable tables and then classify whether an LLM understands these are unionable or not. As future work, we would like to have a framework where an LLM can be a standalone method to perform table-union search.

5 CONCLUSION

We presented ALT-GEN, a framework that uses LLMs to automatically generate tables for the table union search task. Using ALT-GEN, we generated two benchmarks, UGEN-V1 and UGEN-V2, which we showed to be realistic but more challenging benchmarks than existing benchmarks for state-of-the-art table union search methods. Our methodology allows for a more in-depth analysis comparison of union search methods, since it generates labeled non-unionable and allows the user to control the topics, number of topics, and data characteristics used. Our new approach enabled the first topic-based analysis of table union search and the first in-depth analysis of the true/false positive/negative rates for all state-of-the-art methods shedding new light on the strengths and weaknesses of different methods. Finally, we have made both ALT-GEN and benchmarks UGEN-V1 and UGEN-V2 available for other researchers and practitioners to use in their experiments and analysis.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Patricia C. Arocena, Boris Glavic, Radu Ciucanu, and Renée J. Miller. 2015. The iBench Integration Metadata Generator. *PVLDB* 9, 3 (2015), 108–119. https://doi.org/10.14778/2850583.2850586

[2] 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. *PVLDB* 9, 2 (2015), 36–47. https://doi.org/10.14778/2850578.2850579

[3] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. *PVLDB* 17, 2 (oct 2023), 92105. https://doi.org/10.14778/3626292.3626294

[4] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *ICDE*. 709–720.

[5] 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.

[6] Alexander Brinkmann, Roee Shraga, Reng Chiz Der, and Christian Bizer. 2023. Product Information Extraction using ChatGPT. *arXiv preprint arXiv:2306.14921* (2023).

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[8] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying Memorization Across Neural Language Models. In *The Eleventh International Conference on Learning Representations*.

[9] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*. 2633–2650.

[10] Bharath Chintagunta, Namit Katariya, Xavier Amatriain, and Anitha Kannan. 2021. Medically aware GPT-3 as a data generator for medical dialogue summarization. In *Machine Learning for Healthcare Conference*. PMLR, 354–372.

[11] Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling The Internal Knowledge-Base of Language Models. In *Findings of the Association for Computational Linguistics: EACL 2023*. Association for Computational Linguistics, Dubrovnik, Croatia, 1856–1869. https://aclanthology.org/2023.findings-eacl.139

[12] Common Crawl. [n.d.]. Common Crawl. https://commoncrawl.org/.

[13] 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.

[14] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 65–74.

[15] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A Survey for In-context Learning. *arXiv preprint arXiv:2301.00234* (2022).

[16] 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. *PVDLB* 16, 7 (2023), 1726–1739.

[17] Jim Gray (Ed.). 1993. *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*. Morgan Kaufmann.

[18] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. 2016. Schema. org: evolution of structured data on the web. *Commun. ACM* 59, 2 (2016), 44–51.

[19] Mossad Helali, Shubham Vashisth, Philippe Carrier, Katja Hose, and Essam Mansour. 2023. Linked Data Science Powered by Knowledge Graphs. *CoRR* abs/2303.02204 (2023).

[20] 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 *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*. Association for Computational Linguistics, 3786–3800. https://aclanthology.org/2023.findings-acl.233

[21] 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. https://doi.org/10.1145/3588710

[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] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics* 8 (2020), 423–438. https://doi.org/10.1162/tacl_a_00324

[24] 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 *Accepted to appear in SIGMOD Conference*. ACM. https://arxiv.org/pdf/2209.13589.pdf

[25] Aamod Khatiwada, Harsha Kokel, Ibrahim Abdelaziz, Subhajit Chaudhury, Julian Dolby, Oktie Hassanzadeh, Zhenhan Huang, Tejaswini Pedapati, Horst Samulowitz, and Kavitha Srinivas. 2024. TabSketchFM: Sketch-based Tabular Representation Learning for Data Discovery over Data Lakes. *arXiv preprint arXiv:2407.01619* (2024). https://arxiv.org/abs/2407.01619

[26] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *ICDE*. 468–479.

[27] Aristotelis Leventidis, Laura Di Rocco, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2021. DomainNet: Homograph Detection for Data Lake Disambiguation. In *EDBT*, Yannis Velegrakis, Demetris Zeinalipour-Yazti, Panos K. Chrysanthis, and Francesco Guerra (Eds.). 13–24.

[28] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9, Article 195 (jan 2023), 35 pages. https://doi.org/10.1145/3560815

[29] Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. 2018. Table union search on open data. *PVLDB* 11, 7 (2018), 813–825.

[30] OpenAI. 2023. Chat GPT. https://chat.openai.com/chat.

[31] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[32] Koyena Pal, Aamod Khatiwada, Roee Shraga, and Renée J. Miller. 2023. Generative Benchmark Creation for Table Union Search. arXiv:2308.03883 [cs.DB]

[33] David Patterson. 2012. Technical perspective for better or worse, benchmarks shape a field. *Commun. ACM* 55, 7 (2012).

[34] Ralph Peeters and Christian Bizer. 2023. Using chatgpt for entity matching. In *European Conference on Advances in Databases and Information Systems*. Springer, 221–230.

[35] Ralph Peeters and Christian Bizer. 2023. Using ChatGPT for Entity Matching. In *New Trends in Database and Information Systems - ADBIS (Communications in Computer and Information Science)*, Alberto Abelló, Panos Vassiliadis, Oscar Romero, Robert Wrembel, Francesca Bugiotti, Johann Gamper, Genoveva Vargas-Solar, and Ester Zumpano (Eds.), Vol. 1850. Springer, 221–230. https://doi.org/10.1007/978-3-031-42941-5_20

[36] Anna Primpeli and Christian Bizer. 2020. Profiling entity matching benchmark tasks. In *CIKM*. 3101–3108.

[37] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[38] Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Cannim. 2020. Web table retrieval using multimodal deep learning. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 1399–1408.

[39] Kavitha Srinivas, Julian Dolby, Ibrahim Abdelaziz, Oktie Hassanzadeh, Harsha Kokel, Aamod Khatiwada, Tejaswini Pedapati, Subhajit Chaudhury, and Horst Samulowitz. 2023. LakeBench: Benchmarks for Data Discovery over Data Lakes. *arXiv preprint arXiv:2307.04217* (2023).

[40] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

[41] Immanuel Trummer. 2022. CodexDB: Synthesizing code for query processing from natural language instructions using GPT-3 Codex. *PVLDB* 15, 11 (2022), 2921–2928.

[42] Immanuel Trummer. 2022. From BERT to GPT-3 codex: harnessing the potential of very large language models for data management. *PVLDB* 15, 12 (2022), 3770–3773.

[43] Amir Pouran Ben Veyseh, Minh Le Nguyen, Bonan Min, and Thien Huu Nguyen. 2021. Augmenting Open-Domain Event Detection with Synthetic Data from GPT-2. In *ECML/PKDD*.

[44] WDC 2017. T2D Gold Standard. http://webdatacommons.org/webtables/goldstandard.html.

[45] Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2022. Symbolic Knowledge Distillation: from General Language Models to Commonsense Models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 4602–4625. https://doi.org/10.18653/v1/2022.naacl-main.341

[46] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 38–45.

[47] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. 2018. Zero-shot learninga comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence* 41, 9 (2018), 2251–2265.

[48] Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. *arXiv preprint arXiv:2004.11546* (2020).

[49] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Marina Meila and Tong Zhang (Eds.), Vol. 139. PMLR, 12697–12706. https://proceedings.mlr.press/v139/zhao21c.html

[50] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL]

A TECHNICAL REPORT

A.1 Unionability Benchmark Generation

Initial Setup. Before prompting the LLMs about generating the tables, we first populate the topic, shape, and textuality requirements for the table. In our case, we prompt an LLM to generate tables related to different topics in order to create diverse and realistic benchmarks. An important property, which is not present in other benchmarks, is the ability to generate non-unionable tables of the same topic and to let the user of the generator pick the number of topics (meaning how diversity they want their benchmark to be).

Generating Column Headers and the Verification Process. For each topic, we have a query table and a set of data lake tables that are either unionable or not unionable to the former table. We first generate a pair of query and data lake table. As mentioned in Section 3.2, LLMs can be sensitive to the prompts provided since it can interpret the task differently with any subtle variations of the task's description. Hence, while instructing LLMs, we first describe the shape of the tables.

```
Create 2 semi-colon-separated table column headers.
Table 1 has {T1_col} columns on the topic of {T1_topic}.
Table 2 has {T2_col} columns on the same topic.
```

Then, we instruct the LLM about their unionability:

```
They can be unioned because they have {T_unionable_col}
semantically similar columns that can be aligned in
both tables. In other words, {T_unionable_col} of the
columns in table 2 resemble columns in table 1. The
remaining columns don't necessarily resemble any of
the columns in table 1.
```

```
They cannot be unioned because there are no columns in
table 2 that are semantically similar to any columns
in table 1 and vice-versa. In other words, none of the
columns in table 2 resemble any of the columns in table
1.
```

In the prompts, $\{T1_{col}\}$ and $\{T1_{topic}\}$ (and similarly for $T2$) as well as $\{T_{unionable\_col}\}$ are parameters. They may be instantiated randomly during the 'initial setup' phase in Figure 1, but their ranges can also be set by users if a table-pair needs to have certain user-specific conditions. For example,$\{T_{unionable\_col}\}$ represents the number of semantically similar columns that are present between the two tables if they are unionable. To generate benchmarks for evaluating SANTOS [24] we might set the range of this parameter to be from two to ten (since SANTOS requires a minimum of two unionable columns).

To further reduce prompt sensitivity and to allow for increased consistency in the format of the output generated by an LLM, we add the following instruction to the prompt template.

```
Answer the above task in the following format:
Table 1: <table 1-separated table 1 column headers>
Table 2: <table 2-separated column headers>
```

To combat another challenge of LLMs, i.e., hallucination (refer: Section 3.2, we add a verification phase, where we use the response of the LLM from the above prompt, and ask the LLM to verify its' response. Here's an instance of prompt template:

```
Verify that Table 1 has {T1_col} columns and Table 2 has
{T2_col} columns and the generated tables can be unioned.
Re-generate your answer with the corrected response in
the requested format.
```

In the non-unionable case, we change the 'can' to 'cannot' and further clarification that 'none of the columns are semantically similar to any columns in table 1 and vice-versa.'

After verifying the initial query and data lake table pairs for a particular topic, we use the query table column headers and repeat the above process for other data lake tables as illustrated in Figure 2.

Generating Rows. After storing the generated column headers for each table in the benchmark, we populate these tables by prompting the LLM to produce first T_{rows} rows for these tables given their respective column headers. To do so,

```
Given the following column header for a table about
{topic}, generate T_rows table rows where each row has
at least {textuality} words. Here's the column header:
{tableHeaders}. Answer this task in the format of semi-
separated rows, where each row is in a new line.
```

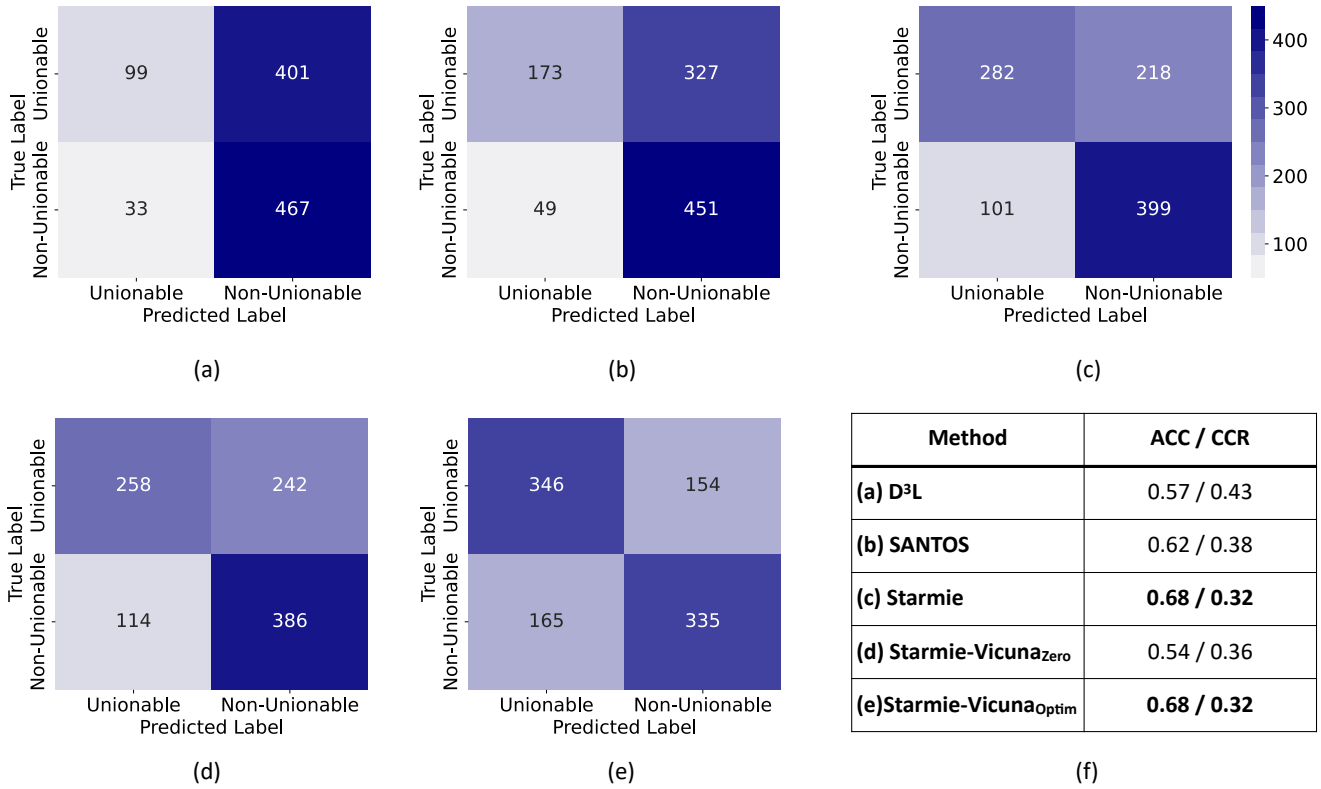
To address scalability, we divide up the task of adding many rows into the previous prompt of generating the first set of rows, and then prompting LLM to add more rows, using previously extracted table headers and a sample of row texts:

```
Given the column headers and last couple rows for a
table about {topic}, generate T_rows more table rows
where each row has at least {textuality} words. Here's
the column header: {tableHeaders}. Here's the last
couple rows:{rowTexts}. Answer this task in the format
of semi-colon-separated rows, where each row is in a
new line.
```

An LLM outputs the requested instructions/prompts as strings. We use post-processing scripts to convert the string-formatted tables into CSV files. In cases were a union search method might require more information than the above tables, we have other scripts to further prompt the LLM and retrieve the required information. For instance, a key or intent column is required by one of the search techniques that we will evaluate [24]. For this, we prompt the LLM to provide us this column based on the query-table and data-lake table pair column headers provided to it.

A.2 Additional Union Search Method Results

Runtime Performance. In Section 3.4, we reported the time and cost (if any) to generate UGEN-V1 and UGEN-V2 benchmarks. The Starmie-LLM variation methods had varying query-time based on their model sizes, length of the prompt, and platform for running them. In the zero-shot case for UGEN-V2, Starmie-GPT-2XL took a total of about 12 minutes, Starmie-Vicuna took about 15 minutes,



Method	ACC / CCR
(a) D³L	0.57 / 0.43
(b) SANTOS	0.62 / 0.38
(c) Starmie	**0.68 / 0.32**
(d) Starmie-Vicuna$_{Zero}$	0.54 / 0.36
(e)Starmie-Vicuna$_{Optim}$	**0.68 / 0.32**

(f)

Figure 6: Confusion matrices for (a) D^{3}L (b) SANTOS (c) Starmie (d) Starmie-Vicuna$_{Zero}$ (e) Starmie-Vicuna$_{Optim}$ methods on distinguishing unionable and non-unionable pairs in UGEN-V2 Benchmark. A legend in the top-right corner is for all the confusion matrices. (f) Accuracy and Corner Case Ratio of different methods in UGEN-V2 Benchmark

and Starmie-Alpaca took around 25 minutes. GPT2-XL is a 1.5 Billion parameter model while Vicuna and Alpaca models are around 7 Billion parameter models. Hence, GPT2-XL had a relatively faster inference time than Vicuna and Alpaca. These query times increase non-linearly. Furthermore, between zero-shot and three-shot, there is about 50% increase in average query time for all models. This is due to the significant increase of the prompt length from a prompt without any examples to one with 3 examples.

Finally,we also create a confusion matrix, we need true labels, false labels, and predicted labels. True labels and false labels are "unionable" and "non-unionable" table pairs from the ground truth respectively. Note that we use search methods that return top-k results for each query. As the total number of unionable table pairs for each query is 10 in the ground truth, we search for the top-10 unionable tables for each query. With 50 query tables, we get 500 predicted true labels. The table pairs that are in the ground truth, but do not make it to the top-10 are false (non-unionable) predicted labels. We compare predicted labels with the ground truth to construct confusion matrices.

We report confusion matrix and accuracy in Figure 6 for each method on UGEN-V2. If we see correct non-unionable predictions (bottom-right of confusion matrices), D^3L seems to be the best at detecting non-unionable tables (predicts 467 out of 500 non-unionable pairs accurately). Notice however, if we look at true positives (top-left of confusion matrix in Figure 6(a)), D^{3}L has the least number, around two times fewer than the second-least performing SANTOS (Figure 6 (b)). This means that the false positives in D^{3}L's results are other unlabeled non-unionable tables from random topics rather than the labeled "non-unionable" tables from the same topic. But if we look at Starmie and Starmie's LLM variations that capture the table context to make unionability decisions, they have high true positives and relatively lower true negatives. This means that their false positives are mostly non-unionable tables from the same topic, as we discussed in Example 1. For instance, Starmie-Vicuna$_{\text{Optim}}$ seems to be the most balanced method in differentiating unionable and non-unionable among the same topic, achieving the highest accuracy. This may be because Starmie captures overall table context well and from the remaining unionable candidates, LLM may also be less confused on the tables from the same topic. This study does reveal that by labeling non-unionable pairs in UGEN-V2, we have been able to create difficult cases that lead to false negatives for all methods. We also get very interesting insight into table unionability – it is important that our benchmarks are not just testing if a method can find tables on the same topic. Instead, the search methods should also separate non-unionability among the same topic tables.