# TabularLens: Leveraging Indexed Retrieval for Table Understanding

Anonymous ACL submission

## Abstract

Advancements in language understanding by Language Models (LMs) have enabled reasoning over tabular data, primarily through training mechanisms that support direct table modification. However, these approaches are often limited to small tables that fit within the model's context window, raising concerns about the scalability of tabular reasoning. To address this challenge, we propose TabularLens, a Retrieval-Augmented Generation (RAG) framework designed to retrieve and structure interpretable table content that can be scaled across multiple tables with different schemas, for LM-based applications. TabularLens employs a two-stage filtering process and a row-column retrieval strategy to efficiently index and extract relevant table elements before passing them to the LM, significantly reducing the input size and enhancing code generation precision. Furthermore, unlike existing models that struggle with proper nouns—such as named entities or domain-specific identifiers—which often lack meaningful embeddings, TabularLens introduces a dedicated mechanism to recognize and appropriately handle such tokens. This ensures robust retrieval and reasoning even when dealing with semantically sparse or opaque table entries.

## 1 Introduction

Large Language Models (LLMs) have significantly advanced the state of natural language understanding and reasoning across diverse domains, including question answering, summarization, and dialogue systems (Brown et al., 2020; Raffel et al., 2023). These models demonstrate strong contextual awareness and logical inference abilities, enabling them to answer complex queries by analyzing both structure and semantics of the input data (Wei et al., 2022; Kojima et al., 2023).

However, as the size and complexity of real-world tasks increase, so do the challenges associated with processing long contexts efficiently (Liu et al., 2024a; Press et al., 2022). Particularly in scenarios where the input consists of extensive textual or structured data, such as documents or tables, naive inclusion of the full context often exceeds model limits, leading to degraded reasoning performance (Beltagy et al., 2020).

Naive approaches to table-based question answering (QA) (Jiang et al., 2023; Yang et al., 2022) often rely on large language models (LLMs) like GPT-4, which use their coding and reasoning abilities to interpret table structures and generate code for data analysis. For small tables, where context length constraints are less prohibitive, it becomes feasible to input the entire table directly into the LLM, allowing it to perform reasoning holistically across the full dataset. However, the same approach cannot be used for larger tables, where including the entire dataset is impractical due to input size limitations. In such cases, LLMs interpret table schemas to generate executable code, but this can lead to loss of critical information present in the table.

To mitigate these issues, retrieval-augmented generation (RAG) frameworks (Lewis et al., 2021) have emerged. These methods decouple retrieval from generation by first identifying the most relevant segments of input before prompting the model (Guu et al., 2020). In the context of table or document understanding, this means selectively extracting schema elements, key-value pairs, or high-salience passages as proposed by (Izacard and Grave, 2021; Borgeaud et al., 2022). By narrowing the context scope to only the most pertinent information, RAG enhances both computational efficiency and reasoning accuracy. Structured data including tables, code, or database schemas, introduces its own constraints making it harder to process using embeddings-based methods.

To overcome all these shortcomings posed, we introduce TabularLens which leverages RAG that employs a multi-layered filtering to enhance at-

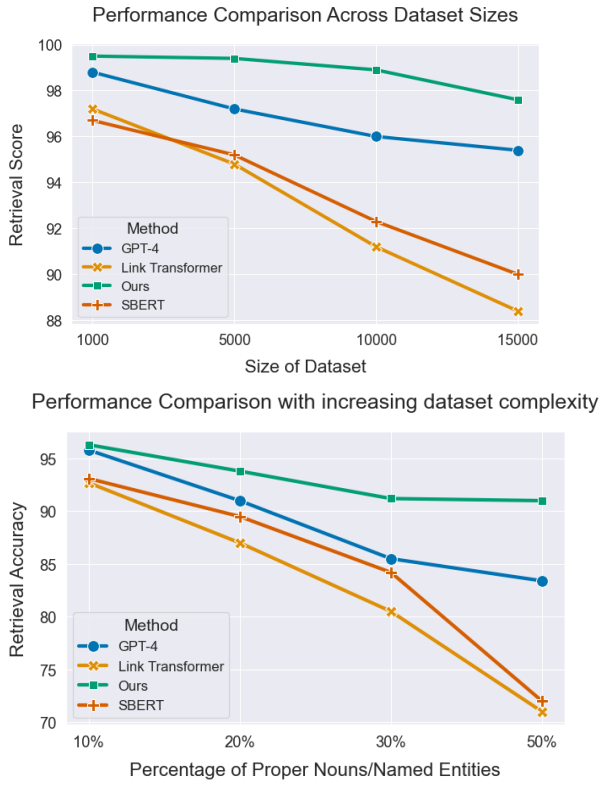


Figure 1: **Top**: Retrieval performance across datasets of varying sizes (row count). **Bottom**: Comparison of TabularLens performance with other methods across datasets of increasing retrieval complexity.

tribute matching for tabular data.

Current approaches pretrain (Herzig et al., 2020; Yin et al., 2020) or utilize either an LLM or embedding models for the row-column extraction, whereas our approach incorporates a row-column retrieval to extract the essential information from the table, which is then passed into an LLM for solving user queries. The retrieval enables the identification of appropriate sub-table and demonstrates superior performance across datasets of varying sizes and complexity as illustrated in Figure 1. Selective columns are retrieved to infer the table schema, which in turn guides the extraction of relevant rows. Furthermore, due to the infeasibility of encoding named entities, we also develop a new algorithm to facilitate their inclusion, which is absent in recent works (Lindemann et al., 2019; Narayan et al., 2017).

They do not account for multi-sourced tables as input, whereas we support merging tables prior to question answering. This is carried out using the record linking algorithm we use for the RAG framework. Hence the retrieval approach utilized in our model provides a solution for multiple applications involving linking records (Christen, 2012).

Our contribution is TabularLens, a novel framework for all-inclusive Tabular RAG. Our novel retrieval algorithm works for all kinds of data types present in the table. We also develop a weighted string matching algorithm that is versetile across several applications.

## 2 Related Works

Research on table understanding has steadily progressed from specialized neural architectures (Herzig et al., 2020; Eisenschlos et al., 2020) to more flexible, few-shot paradigms powered by large language models (LLMs)(Wang et al., 2024; Liu et al., 2024b). Early methods aimed to encode full table structures for tasks such as question answering and semantic parsing. Models like DATER(Ye et al., 2023) and BINDER (Cheng et al., 2023) demonstrated the ability of LLMs to reason over structured data when provided entire tables. However, these approaches are constrained by input length limits and struggle with scalability as table size increases.

To alleviate the reliance on full-table input, two prominent directions have emerged: schema-based generation and retrieval-augmented table reasoning. Schema-based methods, including Text2SQL (Gao et al., 2023) and its successors (Zhong et al., 2017; Sun et al., 2024; Pourreza and Rafiei, 2024, 2023; Wang et al., 2025a), emphasize the structural aspects of tables—treating columns as schemas and generating logical forms such as SQL queries. TabSD (Wang et al., 2025b) proposed a method to decompose tables using SQL commands to removing noise and pre-process sub-tables. (Ji et al., 2024) introduced a hierarchical reasoning framework for large-scale table understanding by performing similar decomposition and organizing information into a tree-structured format.
Conversely, retrieval-based approaches like ITR (Lin et al., 2023) and TAP4LLM (Sui et al., 2024) attempt to scale to larger tables by retrieving salient rows and columns. Although this mitigates context length concerns, these models often depend on static embedding encoders and can underperform in cases involving ambiguous or domain-specific tokens due to limited semantic generalization. Furthermore, a cell retrieval method proposed by TableRAG (Chen et al., 2024) intends to embed each cell of the table and retrieve only the relevant ones. Though this approach may

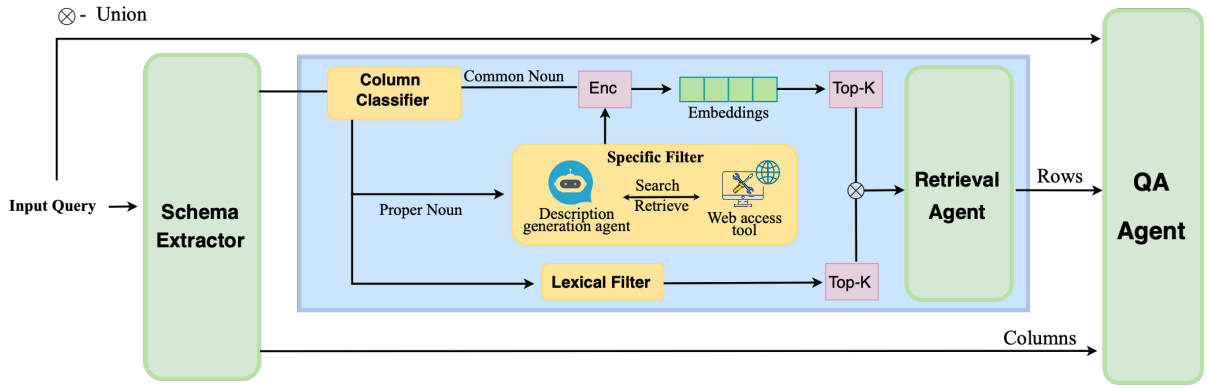


Figure 2: Architecture of TabularLens framework for table understanding. The pipeline illustrates our two-stage filtering approach for tabular reasoning. The Schema Extractor identifies relevant columns from the input query, while the Column Classifier categorizes columns as Generic or Specific. The Top-K condidates from the respective filters are passed into the Retrieval agent for the final stage of matching. A compact sub-table that is passed to the QA Agent for final reasoning.

reduce the inference time for medium sized tables (Typically 50,000 rows), it highly depends upon the embeddings produced and when the number of cells increase in the table the accuracy of the answers drop.

TabularLens distinguishes itself from these prior approaches through several key innovations. Unlike embedding-based methods such as SBERT or standard TableRAG that struggle with proper nouns and domain-specific terminology, TabularLens introduces a specialized mechanism for handling named entities and domain-specific identifiers that traditionally have poor embedding representations. Most notably, TabularLens demonstrates exceptional performance improvements on complex datasets with real-world characteristics, such as AdSpend and HAdSpend, where it achieves gains of 6-8% in accuracy over competing approaches. This performance advantage stems from its ability to integrate multiple record-linking strategies (that is responsible for the retrieval of rows) and handle multi-sourced tables prior to question answering—capabilities absent in most existing frameworks.

## 3 TabularLens

### 3.1 Motivation

Figure 2 illustrates the architecture of our proposed pipeline. The central goal is to integrate named entity and common noun recognition with schema-level retrieval to enable efficient and accurate question answering over tabular data. In most real-world scenarios, processing an entire table is unnecessary and often infeasible due to context length constraints. Instead, extracting a minimal sub-table—comprising only the relevant rows and columns—provides both computational efficiency and interpretability. For example, given a question such as "What is the cost efficiency of advertising on radio?" and a table containing information on various spending channels, it is sufficient to retrieve only the rows referring to radio transmission and the schema elements necessary for computing cost efficiency. Our framework, TabularLens, is specifically designed to address this challenge. It leverages both row and column-level retrieval strategies to identify a compact, semantically rich sub-table, which is then passed to the language model for final reasoning.

Unlike prior approaches that struggle with proper nouns or domain-specific entities—due to poor embedding representations—TabularLens introduces specialized mechanisms for handling named entities. This enables robust retrieval even when tables contain opaque or sparsely annotated tokens. Overall, this motivates a retrieval-augmented approach for table-based question answering.

### 3.2 Problem Formulation

For applications of table understanding the input table $T$ can be represented as an $N \times M$ matrix where $T = t_{ij} \mid 1 \leq i \leq M,\ 1 \leq j \leq M$. Where $N$ is the total number of rows and $M$ is the total number of columns and $t_{ij}$ are the cell values. The

query passed as input can be referred to as $Q$ and the retrieved sub-table as $t_r$. Given the infeasible size of the input table T, we also use a prompt designed to handle the subtable when passed into the Language Model $L$. The response $A = L(P(t_r))$. Our objetive is to optimize the indexing of the subtable $t_r$, making it feasible for the LM to process.

### 3.3 Components of TabularLens

**Schema Extractor** To accurately index the subtable, it is important to identify the relevant columns. We utilize a language model that is prompted to choose which columns and values should be accessed from the input query. A basic embedding model may struggle with more complex questions, such as *What is the cost-efficiency of Company X?* In such cases, cost-efficiency may not be a direct column in the table, and the model must instead infer which columns are related. This response $R$ can be expressed as: $R = C_i, V_i : i < degree(T)$, where $C_i$ are the selected columns, $V_i = v_1, v_2, ..$ are the values referenced, and $degree(T)$ is the number of columns in the input table.

**Column Classifier** To identify relevant rows from the selected columns $C_i$, we match the extracted values $V_i$ against unique entries in the table. Columns are first classified as either *Generic* (common nouns) or *Specific* (proper nouns), and then routed through their respective filters accordingly.

**Generic Filter** Common nouns being abundant can be numerically represented by vanilla embedding models like nomic or SBERT. The Generic filter handles these columns filled with common nouns using an encoder $f_{enc}$, which encodes $Vi$ and compares it against the values in $C_i$. The top match for each value in $Vi$ is then used to filter the rows.

**Specific Filter** To address the semantic uncertainty inherent in proper nouns, we introduce a filtering mechanism within our retrieval pipeline that leverages a language model (LM) augmented with a web-based description retrieval module (This one-time processing step can be performed prior to the QA session for faster inference). When a column predominantly contains proper nouns, direct comparison becomes insufficient due to the subjective and context-dependent nature of these entities. To mitigate this, the LM first identifies proper nouns within the column and issues web queries to retrieve contextual descriptions. These descriptions serve as semantic anchors, enabling the LM to perform more informed comparisons and select the top-K most relevant row values. In cases where the queried proper nouns lack sufficient web presence, the pipeline gracefully degrades to a lexical-based filtering strategy.

**Lexical Filter** we propose a weighted string matching algorithm designed to robustly handle various forms of string ambiguity. This method is particularly effective for proper nouns, where the order of word occurrence plays a critical role in semantic interpretation. As illustrated in Tab. 1, our model demonstrates improved performance in scenarios involving word-order ambiguity, where conventional fuzzy matching techniques commonly employed in retrieval-augmented generation (RAG) pipelines tend to fail.

The algorithm begins by tokenizing each input string $S_i$ into a sequence of words $[w_1, w_2, ...]$. It then computes word-level similarity using a character overlap metric and applies a position-sensitive weighting scheme, which assigns exponentially decreasing importance to successive words. Unlike standard similarity metrics, this approach emphasizes early word matches, reflecting the intuition that leading terms often carry greater semantic weight in named entities.

For each word $w_i$ in the first string $S_1$, we compute an occurrence score $O_i$, defined as the maximum normalized length of continuous character overlap with any word in the second string $S_2$:

$$O_i = \max_{j \in \{1,\ldots,m\}} \left\{ \frac{\text{Continuous Overlap}(w_i, w_j)}{\text{length of } w_i} \right\} \quad (1)$$

The aggregation of the word-level scores for the overall similarity score is then computed as:

$$Score(S_1, S_2) = \sum_{i=1}^{n} W_i \cdot O_i = \sum_{i=1}^{n} 2^{-(i-1)} \cdot O_i \quad (2)$$

$$Score(S_1, S_2) = 1 \cdot O_1 + \frac{1}{2} \cdot O_2 + \ldots + \frac{1}{2^{n-1}} \cdot O_n \quad (3)$$

where, $W_i = 2^{-(i-1)}$. However, since the algorithm is not inherently commutative, we adopt a bidirectional scoring strategy, computing the match in both directions and taking the maximum score as the final similarity value. This modification ensures robustness and practical applicability across a wide range of string comparison tasks involving non-standard or ambiguous entity representations. The final expression can be written as:

$$Score(S_1, S_2) = \max(Score(S_1, S_2),\ Score(S_2, S_1)) \quad (4)$$

| String 1 | String 2 | Fuzzy matching | Our Matching |
|---|---|---|---|
| Stanford University | Cornell University | 0.13 | 0.057 |
| Stanford University | University of Stanford | 0.27 | 0.94 |
| Stanford University | Stanford California | 0.74 | 0.87 |

Table 1: Comparison of string matching techniques. Our technique emphasizes the placement and ordering of words within proper names, leading to more accurate similarity scores. It distinguishes between rephrasings that preserve meaning and those that alter semantic relevance, unlike traditional fuzzy matching which may overestimate similarity based on shared tokens alone.

**Retrieval and QA agent** The retrieval agent operates on the filtered candidates to select the most relevant subset, which is then used to retrieve corresponding rows. After extracting the relevant schema and row segments using the filtering mechanisms described earlier, the final step involves reasoning over the resulting sub-table to generate an answer. To facilitate this, we employ a language model (LM) agent, which combines retrieval-augmented generation with structured execution capabilities. The sub-table $t_r$, composed of selected columns $C_i$ and filtered rows, is passed as part of a structured prompt $P(t_r)$ to the LM. To execute analytical or numerical operations (e.g., computing averages, ratios, or identifying maxima), the QA agent is coupled with a lightweight execution environment. This design ensures robustness across diverse question types.

## 4 Experiments

To evaluate the effectiveness of TabularLens, we conducted extensive experiments aimed at assessing both retrieval accuracy and question-answering performance across multiple datasets. Our evaluation focuses particularly on the framework's ability to handle large tables with complex structures and named entities.

### 4.1 Datasets

Existing TableQA benchmarks such as TabFact (Chen et al., 2020) and WikiTableQA (Kweon et al., 2023) primarily feature small to medium-sized tables that fit within the input limits of most language models. However, they fall short in reflecting the complexities encountered in real-world marketing analytics—particularly in terms of table scale, hierarchical attribute structures, and domain-specific taxonomies. To highlight these limitations and rigorously evaluate model robustness and reasoning under scale and structure, we evaluated the model on two new datasets: **AdSpend** and **Hierarchical AdSpend (HAdSpend)**. These datasets are designed to mimic real world marketing datasets with brand names and categories. Both datasets consist of large tables, with row counts scaling up to 30,000. These characteristics make them uniquely suited for benchmarking retrieval-augmented TableQA agents under complex scenarios.

### 4.2 Experimental Setup

Our experiment uses small LMs like GPT-4o-mini for schema extraction and Qwen2.5-7b for description generation. Other agents like Retrieval agent and QA Agent use larger models like Qwen2.5-32b and GPT-4o. Smaller tasks are equipped with smaller LLMs and tasks requiring more reasoning and precision use larger LLMs. For the top candidates selection we opt for a $K$ value of 5. The parameters that best work for the matching agent are, *temperature* of 0.6, *top_p* of 0.8. All experiments were carried out on an NVIDIA A100 GPU with a max consumption of 20GB .

### 4.3 Sub-table Retrieval Performance

To better understand the retrieval quality of TabularLens compared to baseline approaches, we assessed the recall, precision, and F1 scores for both column and row retrieval across our datasets, as shown in Table 2. The ground truths were extracted from manual annotations of the minimal necessary columns and rows required to answer each question accurately.

For column retrieval on the AdSpend dataset, TabularLens demonstrates superiority, being on par with GPT-4 and yielding high F1 score. In contrast, embedding-based approaches showed lower precision, suggesting they retrieved more irrelevant columns. The advantages of TabularLens become even more evident in row retrieval tasks. Our approach achieved near-perfect metrics. This significant improvement in row retrieval performance can be attributed to our hybrid approach that effectively handles both generic and specific columns.

Table 2: Sub-Table Retrieval Metrics (P: Precision, R: Recall, F1: F1 Score). LT - Link Transformer (Arora and Dell, 2023), SBERT (Devlin et al., 2019).

| Model | AdSpend | | | | | | HAdSpend | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Col Retrieval | | | Row Retrieval | | | Col Retrieval | | | Row Retrieval | | |
| | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| Fuzzy Dist. | - | - | - | 0.617 | 0.644 | 0.630 | - | - | - | 0.610 | 0.580 | 0.594 |
| LT | 0.905 | 0.943 | 0.923 | 0.960 | 0.691 | 0.830 | 0.821 | 0.845 | 0.833 | 0.710 | 0.702 | 0.706 |
| GPT-4 | **0.980** | **1.000** | **0.989** | 0.941 | 0.850 | 0.893 | **0.930** | **0.960** | **0.945** | 0.910 | 0.730 | 0.810 |
| SBERT | 0.910 | 0.950 | 0.930 | 0.941 | 0.735 | 0.825 | 0.820 | 0.860 | 0.839 | 0.715 | 0.720 | 0.720 |
| Ours | 0.960 | **1.000** | 0.979 | **0.995** | **1.000** | **0.997** | **0.930** | 0.940 | 0.935 | **0.981** | **0.987** | **0.984** |

Table 3: Accuracy metrics across Tabular Question Answering datasets

| Models | AdSpend | HAdSpend | TabFact | WikiTableQA |
|---|---|---|---|---|
| TableRAG (GPT-3.5-Turbo) (Chen et al., 2024) | 63.1 | 60.6 | 90.1 | **57.03** |
| GPT-4 (ReadTable) | 51.5 | 42.4 | 89.2 | 52.3 |
| GPT-4 (ReadSchema) | 54.5 | 41.4 | 87.5 | 50.6 |
| GPT-4 (RandRowSampling) | 36.4 | 33.7 | 86.2 | 51.9 |
| **Ours(GPT-3.5-Turbo)** | 69.5 | 65.0 | 91.7 | 56.75 |
| **Ours** | **69.7** | **66.6** | **91.7** | 57.0 |

In summary, this analysis underscores TabularLens's efficacy in retrieving essential information in both column and row aspects, particularly when dealing with large tables containing named entities and domain-specific identifiers.

### 4.4 Question Answering Performance

To evaluate our performance on the TableQA task, we compared TabularLens across various datasets, with different table sizes against methods that rely on different kinds of input used for QA purpose.

Following (Chen et al., 2024), we evaluate our approach against three established methods for primary table prompting:

**Read Table**: This method presents the entire table as input to the LLM, allowing complete access to all tabular data at the cost of increased token consumption.

**Read Schema**: This approach provides only the table schema (column headers and data types) as input to the LLM. The model is then prompted to generate code that can selectively retrieve relevant portions of the table required for reasoning, optimizing token efficiency at the potential expense of comprehensive data access.

**RandRow Sampling**: This method is a prevalent rule-based sampling approach (Sui et al., 2024) that randomly selects rows from the table with equal probabilities. When the total number of rows exceeds K, we select K rows to form a representative sample. This baseline is employed to underscore the benefits of more targeted retrieval methods, illustrating how they can provide more relevant and efficient data selection compared to random sampling.

These baseline approaches represent the current standard methodologies against which we benchmark our novel contribution.

As shows in Table3 our approach surpasses other methods across most datasets. Metrics on publicly available datasets like TabFact and WikiTableQA are on par or slightly better than existing approaches due to the simple questions and small-sized tables. Whereas, TabularLens achieves a greater leap in accuracy when tested against dataset with real-time complexities and large and complex table structures.

### 4.5 Ablation

Since TabularLens employs a multi-stage retrieval pipeline with several filtering mechanisms operating in a sequential manner, we conducted comprehensive ablation studies to quantify the contribution of each component to the overall system performance. These experiments serve multiple purposes: (1) identifying which filtering strategies are most effective for different data characteristics, (2) determining the optimal combination of filters for various table structures, and (3) understanding the relative importance of schema extraction versus row retrieval in the end-to-end question answering process.

We systematically evaluated different configurations of our framework by selectively enabling

or disabling specific components, measuring the impact on downstream QA accuracy. This analysis provides insights into the strengths and limitations of each module, particularly in handling tables with complex schemas and named entities. Additionally, we investigate how different model choices and hyperparameter settings affect overall system performance across our benchmark datasets. All ablation results were obtained from testing on the datasets AdSpend and HAdSpend.

| Model Variant | AdSpend | HAdSpend |
|---|---|---|
| *Single Filter* | | |
| Only FM | 31.5 | 28.0 |
| Only LF | 58.6 | 56.5 |
| Only Description | **65.4** | **61.3** |
| *Dual Filters* | | |
| FM + Description | 66.5 | 61.3 |
| LF + Description | **69.7** | **66.6** |

Table 4: Accuracy values for ablation study on different filters used in TabularLens with the AdSpend and HAdSpend datasets. FM - Fuzzy Matching, LF - Lexical Filter.

#### 4.5.1 Filtering Step

To analyze the influence of each filter that contributes to selecting the top candidates for each input, we conducted experiments employing different methodological combinations as shown in Table4. Under the *Single Filter* configuration, our evaluation reveals that the description generator tool, which enables the language model to retrieve contextual information about named entities, delivers the strongest performance across both datasets. This superior performance can be attributed to the prevalence of domain-specific abbreviations and semantic ambiguities in column values that challenge traditional string matching approaches.

When examining *Dual Filter* configurations, we observe that combining fuzzy matching with description generation yields only marginal improvements over using description generation alone. However, the integration of our Lexical Filter with the description generation module produces a substantial performance boost across both datasets. This improvement underscores the complementary nature of these two approaches—the Lexical Filter's position-sensitive weighting scheme effectively captures word-order importance in named entities, while the description generation module provides semantic context for ambiguous terms commonly found in marketing datasets.

#### 4.5.2 Matching Step

The final stage before sub-table querying the QA Agent involves a language model-based matching agent that aligns user input with candidate records. This step is critical, as it governs the accuracy of the entire pipeline. Effective retrieval of the correct record from the table is essential to avoid hallucinated or semantically incorrect responses. Three parameters have a significant influence on matching efficacy: (1) the Top-K candidates chosen from each filter, (2) the Batch size of input passed to the LM, and (3) the choice of the language model employed for semantic alignment.

As shown in Table 5, our proposed configuration—with $K = 5$ and a batch size of 1 —yields the highest average row retrieval accuracy across both the AdSpend and HAdSpend datasets. Ablation across these hyperparameters reveals important trends. While larger Top-K values (e.g., $K = 10$) intuitively provide more candidates for matching, they often introduce noise, particularly in models with limited contextual discrimination capabilities. Conversely, smaller K values improve precision at the cost of potentially excluding relevant rows.

Interestingly, model-specific behavior also plays a role. GPT-4 variants demonstrate stability across input configurations, with minimal degradation in matching performance as batch size scales (atmost 3.5%). This robustness is attributed to their deeper contextual awareness and refined token prioritization. On the other hand, Qwen2.5-32B exhibits superior performance in low-input settings but shows significant performance deterioration with larger K and batch size (atmost 8%), likely due to context overflow and overfitting to irrelevant tokens. DeepSeek-32B, despite excelling in tasks such as mathematical and philosophical reasoning, struggles in tasks like table alignment. Its verbose reasoning output leads to high inference latency and increased token bloat, both of which compromise precision and induce erroneous assumptions.

In conclusion, our ablation confirms that careful tuning of these parameters, aligned with model-specific strengths, is imperative for optimizing retrieval and minimizing semantic drift

| Num Input Set | GPT-4o-mini | | GPT-4o | | DeepSeek-32B | | Ours(Qwen2.5-32B) | |
|---|---|---|---|---|---|---|---|---|
| | $K = 5$ | $K = 10$ | $K = 5$ | $K = 10$ | $K = 5$ | $K = 10$ | $K = 5$ | $K = 10$ |
| 1 | 92.2 | 90.1 | **97.5** | 94.2 | 89.3 | 89.1 | **99.2** | 96.4 |
| 2 | 92.2 | 88.5 | 96.0 | 94.0 | 88.5 | 87.7 | 97.3 | 94.2 |
| 5 | 91.5 | 87.4 | 95.5 | 92.5 | 86.2 | 85.2 | 93.2 | 91.8 |
| 8 | 88.6 | 87.0 | 93.6 | 91.3 | 84.7 | 82.4 | 91.1 | 88.8 |

Table 5: Effect of varying Top-K candidates and matching query batch size on matching across different LLMs. **Red** represents the model that attains the highest accuracy followed by **Blue**

in large-scale table-based QA.

## 5 Conclusion

In this work, we presented TabularLens, a robust retrieval-augmented framework tailored for scalable table-based question answering. By integrating schema-aware column extraction, named-entity sensitive row filtering, and a hybrid lexical-semantic matching strategy, TabularLens demonstrates strong performance across diverse datasets—particularly excelling in scenarios involving large, multi-source tables and domain-specific terminology.

Our experiments confirm that TabularLens not only improves retrieval fidelity but also leads to meaningful gains in downstream question-answering accuracy, even outperforming models that ingest full tables. The framework's modular design, with dedicated components for handling both common and proper nouns, offers flexibility across use cases and opens up possibilities for broader table-centric reasoning applications.

## 6 Limitations and Future Work

While TabularLens demonstrates strong performance and scalability, there remain several avenues for enhancement and extension. One promising direction is automating schema evolution, enabling dynamic adaptation to evolving table structures without manual intervention. Additionally, the current framework relies on predefined thresholds and filtering heuristics; integrating learned ranking models or reinforcement learning-based selection policies could further optimize retrieval quality.

That said, a few aspects merit further exploration. While our description-based filtering mechanism effectively boosts retrieval for named entities, it introduces latency due to web-based lookups during preprocessing. This can be handled by retreiving the description prior to passing in the data through the pipeline, in this way the inference is faster but demands precomputation of the descriptions. Additionally, the performance of our retrieval module may vary across domains where external descriptions are scarce or inconsistent. Finally, although the system generalizes well across marketing datasets, evaluating its robustness across other structured domains like healthcare or finance would further validate its applicability.

We believe addressing these limitations and expanding TabularLens into broader, domain-agnostic applications will enhance the reliability and versatility of retrieval-augmented reasoning in structured data environments. Overall, TabularLens moves a step closer to enabling efficient and interpretable table understanding at scale. We hope this work encourages further research into retrieval-based reasoning strategies for structured data.

## References


Abhishek Arora and Melissa Dell. 2023. Linktransformer: A unified package for record linkage with transformer language models. *Preprint*, arXiv:2309.00789.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. *Preprint*, arXiv:2112.04426.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024. Tablerag: Million-token table understanding with language models. *Preprint*, arXiv:2410.04739.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Peter Christen. 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Publishing Company, Incorporated.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *Preprint*, arXiv:2308.15363.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. *Preprint*, arXiv:2007.01282.

Deyi Ji, Lanyun Zhu, Siqi Gao, Peng Xu, Hongtao Lu, Jieping Ye, and Feng Zhao. 2024. Tree-of-table: Unleashing the power of llms for enhanced large-scale table understanding. *Preprint*, arXiv:2411.08516.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Sunjun Kweon, Yeonsu Kwon, Seonhee Cho, Yohan Jo, and Edward Choi. 2023. Open-wikitable: Dataset for open domain question answering with complex reasoning over table. *Preprint*, arXiv:2305.07288.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. An inner table retriever for robust table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926, Toronto, Canada. Association for Computational Linguistics.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024b. Rethinking tabular data understanding with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 450–482, Mexico City, Mexico. Association for Computational Linguistics.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 606–616, Copenhagen, Denmark. Association for Computational Linguistics.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Preprint*, arXiv:2304.11015.

Mohammadreza Pourreza and Davood Rafiei. 2024. DTS-SQL: Decomposed text-to-SQL with small large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8212–8220, Miami, Florida, USA. Association for Computational Linguistics.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2024. Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. *Preprint*, arXiv:2312.09039.

Ruoxi Sun, Sercan Ö. Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, and Tomas Pfister. 2024. Sql-palm: Improved large language model adaptation for text-to-sql (extended). *Preprint*, arXiv:2306.00739.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025a. Mac-sql: A multi-agent collaborative framework for text-to-sql. *Preprint*, arXiv:2312.11242.

Yuxiang Wang, Junhao Gan, and Jianzhong Qi. 2025b. Tabsd: Large free-form table question answering with sql-based table decomposition. *Preprint*, arXiv:2502.13422.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. TableFormer: Robust transformer modeling for table-text encoding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 528–537, Dublin, Ireland. Association for Computational Linguistics.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning. *arXiv preprint arXiv:2301.13808*.

Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *Preprint*, arXiv:2005.08314.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *Preprint*, arXiv:1709.00103.