

SODIUM: From Open Web Data to Queryable Databases

Chuxuan Hu
UIUC
chuxuan3@illinois.edu

Maxwell Yang
UIUC
my37@illinois.edu

Philip Li
UIUC
philipl2@illinois.edu

Daniel Kang
UIUC
ddkang@illinois.edu

ABSTRACT

During research, domain experts often ask analytical questions whose answers require integrating data from a wide range of web sources. Thus, they must spend substantial effort searching, extracting, and organizing raw data before analysis can begin. We formalize this process as the SODIUM task, where we conceptualize open domains such as the web as latent databases that must be systematically instantiated to support downstream querying. Solving SODIUM requires (1) conducting in-depth and specialized exploration of the open web, which is further strengthened by (2) exploiting structural correlations for systematic information extraction and (3) integrating collected information into coherent, queryable database instances.

To quantify the challenges in automating SODIUM, we construct SODIUMBENCH, a benchmark of 105 tasks derived from published academic papers across 6 domains, where systems are tasked with exploring the open web to collect and aggregate data from diverse sources into structured tables. Existing systems struggle with SODIUM tasks: we evaluate 6 advanced AI agents on SODIUMBENCH, with the strongest baseline achieving only 46.5% accuracy. To bridge this gap, we develop SODIUMAGENT, a multi-agent system composed of a web explorer and a cache manager. Powered by our proposed ATP-BFS algorithm and optimized through principled management of cached sources and navigation paths, SODIUMAGENT conducts deep and comprehensive web exploration and performs structurally coherent information extraction. SODIUMAGENT achieves 91.1% accuracy on SODIUMBENCH, outperforming the strongest baseline by approximately 2 \times and the weakest by up to 73 \times .

PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at <https://github.com/uiuc-kang-lab/sodium>.

1 INTRODUCTION

Domain experts decompose their research objectives into smaller analytical subproblems. At each stage, they formulate a concrete analytical question together with a provisional database schema. However, the corresponding values are often scattered across deeply nested webpages within specialized websites. For example, demographic research frequently investigates questions whose underlying data are publicly available on government statistical portals [11, 14, 31, 37–39, 49, 61–63]. To study “disability-free grandparenthood in Italy between 1998 and 2016” (Q_1) [38], a researcher defines a table with primary key year (1998–2016) and columns representing household metrics. Yet the required values are distributed across

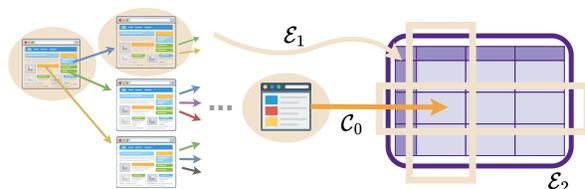


Figure 1: Solving SODIUM is driven by (C_0) in-depth exploration of specialized websites, and strengthened through (E_1) exploiting structural correlations for systematic information extraction and (E_2) integrating collected information into coherent, queryable database instances.

multiple datasets and webpages. Before any analysis can proceed, the researcher needs to navigate the Italian National Institute of Statistics (Istat) website [4] to access data from the Family and Social Subjects (FSS) surveys of different years, as well as the corresponding Italian life tables, and manually aggregate them into a unified table. As a result, substantial overhead effort is spent on searching, extracting, and organizing raw values prior to performing the intended analytical task [12, 24, 59]. Given the rapid progress of large language models (LLMs) and AI agents on complex reasoning and interaction tasks [2, 23, 36, 44, 56], automating this process becomes both feasible and compelling.

We formalize this process as the SODIUM task, which focuses on Structuring Open Domain Unstructured Data into Materialized Databases, in Section 2. In SODIUM, users pose a concrete analytical query over a particular organization’s or platform’s official website and specify the expected database schema. The system is required to populate this given schema with values discovered from the open web. SODIUM models open domains such as the web as latent databases and formalizes the process of systematically materializing a database instance. As we show in Figure 1, solving a SODIUM task relies on a fundamental open-domain search capability (C_0), complemented by two structure-aware extensions that enable cross-cell reasoning and coherent database materialization (E_1 , E_2):

(C_0) In-depth information discovery. A system must go beyond surface-level page inspection and conduct hierarchical, multi-step navigation to locate fine-grained data embedded within complex web portals. For example, to retrieve the household data needed for Q_1 , one must first locate the Istat data exploration portal, from which it selects the “Population and Households” category among 20 available options, then navigate to the “Households” subcategory

among 10 subcategories, and finally identify the correct “Type of Households” dataset among 22 datasets.

(E1) Identification and utilization of structural correlations across cells. Webpages often follow consistent organizational patterns, where relationships across entities (columns) and primary keys (rows) can be mapped to regular navigation patterns across webpages. Thus, structural dependencies across table cells can guide the filling of related entries by aligning web structures with table structures. For example, after retrieving a value under the “Type of Households” dataset in Q_1 , locating values under the “Household Size” dataset does not require restarting the search process; instead, the system can directly navigate to the corresponding “Households” subcategory page by reusing the previously identified organizational structure (*column dependency*). Similarly, once values for the year 1998 are retrieved, the remaining years are located in structurally similar webpages (*row dependency*). Exploiting these structural regularities optimizes exploration and supports scalable, efficient, and systematic information extraction.

(E2) Organization of collected information into structured, queryable databases. The populated table serves as input for downstream analysis and must therefore maintain internal consistency. For example, in Q_1 , the datasets on Istat provide two types of measures (“Thousands value” and “Per 100 households with the same characteristics”), and the system must ensure consistent formatting and representation across these measures.

To systematically evaluate SODIUM, we construct SODIUMBENCH, a benchmark of 105 analytical queries together with their target table schema and specialized domains derived from 48 published academic papers spanning 6 domains (Section 3). The scale and diversity of SODIUMBENCH underscore that structuring open-domain web data into materialized databases is a widespread requirement in practical research workflows. In each SODIUMBENCH task, systems are tasked with exploring the open web in the specialized domain to populate the table based on the query.

We evaluate 6 state-of-the-art agents with the most advanced web search capabilities on SODIUMBENCH in Section 5. Our evaluation shows that they perform poorly on SODIUM tasks, with AG2 [2] achieving the highest accuracy at 46.5%, and Open Deep Research [43] achieving the lowest at 1.2%. These results empirically demonstrate that existing systems do not align with the core demands of SODIUM. In fact, none of the existing systems fully realizes any of C_0 , \mathcal{E}_1 , or \mathcal{E}_2 , as we elaborate through the following detailed analysis of the systems with the closest related functionality.

First, while existing web search engines [5], LLM-integrated web search tools [8–10], and web agents [25] have shown promising performance [30], they focus on general-purpose and surface-level exploration of the open web. In contrast, C_0 requires comprehensive, in-depth explorations of specialized web domains. With the exception of WebVoyager, which attempts to enhance exploration but ultimately traverses only a narrow subset of available navigation paths and achieves less than 3% accuracy on SODIUMBENCH, existing systems and agent frameworks [2, 36, 56] do not optimize at the search tool level, which fundamentally limits their ability to perform deep, domain-aware exploration.

Second, existing information retrieval (IR) processes, including the retrieval phase of RAG-based systems [22] and even table-population workflows [68], treat each retrieval independently, ignoring structural dependencies across related searches. Although the underlying data pool exhibits internal structure (e.g., graph [26]), these systems fail to link related retrievals by exploiting shared structural regularities, as embodied in \mathcal{E}_1 .

Third, existing LLM-based data management systems, whether operating over open-domain [28, 70] or local data [27, 32], assume that the input data is already organized and well structured. Thus, they bypass the central challenge addressed by \mathcal{E}_2 : guaranteeing the coherence and consistency during materializing databases.

To overcome the limitations of existing systems, we develop SODIUMAGENT, an agentic system that explores open domains to populate structured tables in Section 4. SODIUMAGENT consists of two integrated core components: a web explorer and a cache manager. We design a novel ATP-BFS algorithm as the exploration workflow for the web explorer to conduct deep, comprehensive, and domain-specialized navigation of the open web. The cache manager stores and reuses discovered sources and navigated search paths to reduce redundant explorations and preserve coherence across related table cells.

We evaluate SODIUMAGENT on SODIUMBENCH in Section 5. Our results show that SODIUMAGENT achieves 91.1% task accuracy, outperforming the strongest baseline by approximately $2\times$ and the weakest by up to $73\times$ at the task level. We also demonstrate SODIUMAGENT’s superiority across different dimensions of SODIUM, with particularly pronounced gains from the structured database perspective (\mathcal{E}_2) achieving at least a $16.6\times$ improvement.

In summary, our contributions are threefold:

- (1) We formalize the SODIUM task, which materializes open web data into structured databases for downstream analysis.
- (2) We construct SODIUMBENCH, a benchmark that quantifies the challenges of SODIUM and systematically reveals the limitations of existing AI agents.
- (3) We develop SODIUMAGENT, an agentic system that performs in-depth and comprehensive web exploration through our proposed ATP-BFS algorithm and manages navigation sources and paths in a principled manner for scalable exploration, achieving over 90% accuracy on SODIUMBENCH.

2 SODIUM

In this section, we first introduce real-world applications, i.e., the motivations for SODIUM in Section 2.1, and then formally define the task in Section 2.2.

2.1 Background and Use Cases

Researchers in scientific domains decompose their overall research objectives into analytical subproblems. At each stage, they start with precise analytical questions and an ideal database schema, but must first collect and aggregate data from a wide range of web sources into structured tables before any analysis can be conducted. This is particularly common in fields such as demographics [11, 14, 31, 37–39, 49, 61–63], economics [52, 53, 57], sports [13, 15, 18, 19, 42, 50, 64], and many other data-driven fields that use authoritative statistics published online by government agencies or international

organizations. However, the required data are often distributed across deeply nested webpages, multiple reports, or separate annual releases, requiring substantial manual effort to collect and organize.

For example, Table 1 in Treleaven and Banchoff [61] presents demographic and health survey (DHS) statistics from 2000 to 2020 across multiple sub-Saharan African countries. The table uses a relational structure with country as the primary key, where each row corresponds to a specific country and each column represents demographic attributes, including the number of surveys conducted, survey years, the number of children under five, and the number of women of reproductive age. To construct this table, a key step is to populate the attributes for each country by systematically exploring the DHS website.¹ The homepage contains diverse and heterogeneous content, including announcements, reports, and multiple navigation tabs (e.g., *Countries*, *Data*, *Publications*, *Methodology*, *Research*, and *Topics*), many of which appear potentially relevant. Selecting the correct navigation path requires reasoning about the intended table schema: specifically, recognizing that the primary key is *country* and first navigating to the *Countries* section², where individual country pages are listed. From there, the researcher selects the country to investigate, e.g., Benin.³

Similarly, Table 2 in Santos et al. [51] uses swim events from the 2022 FINA World Championships as the primary keys, where each row corresponds to a specific event and the columns represent swimmer statistics across different competition stages (e.g., top 16, semi-finalists, and finalists). The data collection procedure is structurally similar to the previous example, requiring researchers to explore the official FINA website.⁴ However, in this case, different attributes for the same primary key are located on separate webpages, introducing additional complexity. For example, the statistics for final round⁵ of the event *Women 50m Freestyle* are located at a different page than the semi-finalists of the same event.⁶

These examples illustrate that populating such tables requires **in-depth exploration of the open web**, involving complex and multi-step navigation.

Importantly, researchers do not restart the entire process for every primary key. In the DHS example, after collecting data for Benin and proceeding to the next country, Burundi, they do not return to the homepage. Instead, they navigate back to the *Countries* section and directly select Burundi’s page⁷ from there. In other words, they **use structural features of the webpages and previously discovered navigation paths to optimize their explorations**.

Finally, because the collected data are intended for downstream analysis, researchers must **ensure the table has consistent and coherent formats**. For example, time measurements extracted from the FINA website must be normalized into a unified unit, such as seconds, and displayed in a consistent *XX:XX* format.

¹<https://www.dhsprogram.com/>

²<https://www.dhsprogram.com/Countries/>

³https://www.dhsprogram.com/countries/Country-Main.cfm?ctry_id=52

⁴<https://www.worldaquatics.com/>

⁵<https://www.worldaquatics.com/competitions/2894/16th-fina-world-swimming-championships-25m-2022/results?event=4e32ebf1-b383-4acf-87b6-8f2349da6b33&unit=final>

⁶<https://www.worldaquatics.com/competitions/2894/16th-fina-world-swimming-championships-25m-2022/results?event=4e32ebf1-b383-4acf-87b6-8f2349da6b33&unit=semifinals-summary>

⁷https://www.dhsprogram.com/countries/Country-Main.cfm?ctry_id=51

Together, these examples reveal a recurring workflow: researchers begin with a target schema aligned with their analytical queries, but the corresponding values are scattered across multiple web sources. Apart from the analytical computation itself, a substantial portion of the effort also lies in discovering, extracting, normalizing, and materializing the required data into a structured database instance. This gap between analytical intent and fragmented, deep web data motivates the SODIUM problem.

2.2 Problem Definition

We formally define the SODIUM problem as follows:

Input: (1) an analytical query q in natural language, (2) a base domain u (e.g., a website or domain scope like the DHS and FINA homepage in the examples in Section 2.1), and (3) a target relational schema \mathcal{T} specifying a designated primary key and a set of attributes, together with a set of primary key values defining the rows to be populated.

Task: Discover and structure relevant information from open-domain web sources within domain u to populate the schema \mathcal{T} .

Output: A materialized table instance \mathcal{I} conforming to schema \mathcal{T} , where each cell value $\mathcal{I}[r, c]$ is instantiated with values extracted from the open web.

3 SODIUMBENCH

Existing benchmarks do not capture the requirements of SODIUM tasks. Traditional table-filling benchmarks [68], information retrieval benchmarks [60, 69], data science benchmarks [27, 32], and NL2SQL benchmarks [33, 35] assume that relevant documents or databases are locally available and well-structured. Existing open-domain search benchmarks either focus on retrieving simple and independent textual content [25] or assume that structured databases have already been instantiated online [28].

To systematically evaluate the SODIUM problem in real-world settings, we construct SODIUMBENCH, a benchmark derived from published academic papers across diverse scientific domains.

To ensure that SODIUMBENCH reflects realistic and non-trivial data collection tasks, we apply the following selection criteria:

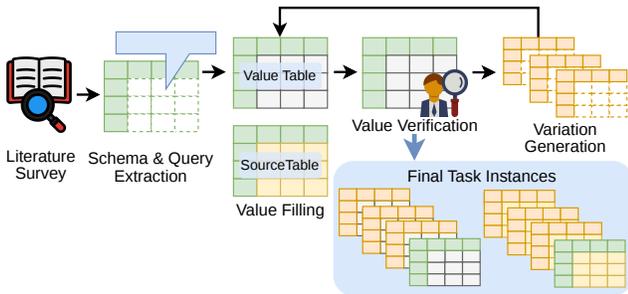
- (1) The study requires data that can be structured into a relational table with unambiguous primary keys and attributes.
- (2) The required data for each study must span multiple webpages (i.e., cannot be retrieved from a single page).
- (3) All data must be directly extractable from webpages without complex offline processing or heavy data transformation.
- (4) All relevant webpages must be publicly accessible and must not require authentication or restricted access.

As we document in Figure 2, our data collection process goes through a systematic workflow composed of the following stages.

Literature Survey. To ensure (i) ecological validity, we derive tasks from published research papers across 6 common domains; (ii) recency and timeliness, we collect papers up to a unified cutoff date of December 2025; (iii) sufficient task diversity and balanced coverage across fields, we determine domain-specific starting years based on two considerations: (1) the publication frequency and overall volume of papers in the venue, and (2) the proportion of recent papers that satisfy our task selection criteria. Specifically, we examine

Table 1: Statistics of SODIUMBENCH.

Domain	# Surveys	# Tables	# Base Domains	# Primary Keys	# Cols	# Cells	Avg. # Search Depth
Demographics	8	8	7	29	49	188	3.75
Sports	7	30	9	162	184	885	4.35
Finance	11	22	10	99	85	379	3.33
Economics	5	12	6	46	41	154	3.39
Food	9	19	9	75	90	354	3.12
Climate	8	14	8	47	57	189	3.08
Total	48	105	49	458	506	2149	3.81

**Figure 2: Data collection workflow of SODIUMBENCH.**

the most recent issues of each venue to estimate the percentage of papers, denoted by α , whose underlying data meet our eligibility requirements. Because this proportion varies substantially across domains (e.g., many sports studies rely on interview-based or restricted data), whereas economics studies mostly use publicly accessible datasets), earlier starting years are required in domains with lower α to obtain a sufficient number of eligible tasks. We select starting years such that approximately $10/\alpha$ papers are surveyed per domain, yielding roughly 10 qualifying tasks in expectation:

- **Demographics:** Papers published from September 2023 to December 2025 in *Demographic Research*.⁸
- **Sports:** Papers published from 2021 to 2025 in *Journal of Sports Science & Medicine*.⁹
- **Finance:** Papers published in 2025 in *Journal of Finance*.¹⁰
- **Economics:** Papers published in 2025 by the *American Institute for Economic Research*.¹¹
- **Food:** Papers published in 2025 in *Nature Food*.¹²
- **Climate:** Papers published from August 2025 to December 2025 in *Nature Climate Change*.¹³

Overall, we reviewed 1,004 papers and identified 48 that satisfy our task construction criteria, with the domain-wise distribution documented in Table 1.

Schema and Query Generation. For each reviewed paper, we construct table schema directly from tables or figures that summarize collected data (e.g., metadata tables, cross-country comparisons, or

survey statistics) when such structured representations are available. In cases where no explicit table or figure is provided, we derive the schema by summarizing the data collection process described in the text, ensuring that the resulting relational structure satisfies our selection criteria while faithfully reflecting the underlying data analytical workflow. Each resulting task instance consists of:

- a natural language analytical query q describing the research objective of the paper,
- a base domain u specifying the authoritative data source (i.e., the official homepage of the relevant organization or data portal);
- a target relational schema \mathcal{T} specifying a designated primary key and a set of attributes, together with a set of primary key values defining the rows to be instantiated.

The objective is to construct a materialized table instance over \mathcal{T} by systematically retrieving and structuring values from diverse open-domain web sources.

Value Collection. A team of two annotators is responsible for instantiating the table values using only the query, schema, and base domain (i.e., the same inputs provided to evaluated systems). Annotators independently retrieve and fill in all cell values from public web sources, recording both the materialized table and the source URLs.

Value Verification. To ensure correctness and reliability, a team of four annotators verifies the collected values. Verification involves comparing reconstructed tables against the original paper tables whenever possible. In cases where discrepancies arise (e.g., evolving public datasets), annotators resolve conflicts until full mutual agreement based on documented web evidence.

Variation Generation. To increase robustness and evaluation coverage, we generate additional variations within the same base domain and difficulty level. For example, if an original study analyzes results from a *Female Super Sevens Tournament* [15], we construct a parallel task for the corresponding *male* tournament. All variations then undergo the same verification procedure, as previously described.

Following this workflow, we construct 105 tasks from 48 published studies across 6 scientific domains. We provide detailed statistics of SODIUMBENCH in Table 1. In total, SODIUMBENCH contains 458 primary key values and 506 attributes, resulting in 2149 cell values that must be instantiated from distributed web sources (i.e., 2149 retrieval operations are required). The scale, diversity, and structural complexity of SODIUMBENCH reflect realistic database

⁸<https://www.demographic-research.org/>

⁹<https://www.jssm.org/>

¹⁰<https://onlinelibrary.wiley.com/journal/15406261>

¹¹<https://aier.org/features/paper/>

¹²<https://www.nature.com/natfood/>

¹³<https://www.nature.com/nclimate/>

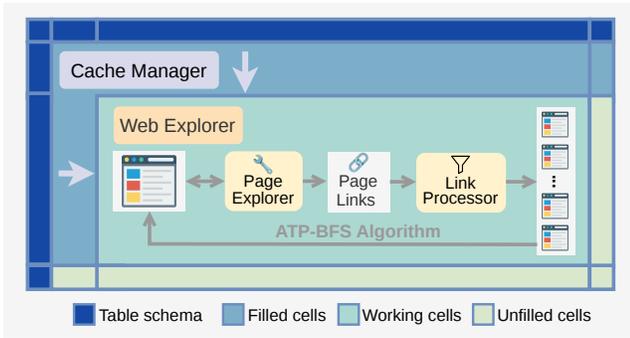


Figure 3: Overview of SODIUMAGENT.

materialization workloads observed in domain research, establishing it as a rigorous and representative benchmark for evaluating the SODIUM problem.

4 SODIUMAGENT

In this section, we introduce SODIUMAGENT, an agentic system that aggregates and structures web information. Given a query, a base domain, and an output table schema defined by its primary keys (rows) and attributes (columns), SODIUMAGENT fills in each cell in sequence. As we illustrate in Figure 3, SODIUMAGENT consists of two core components: (1) a web explorer (Section 4.1), which performs in-depth exploration to extract cell-specific information from web pages and equips the agent with mechanisms to construct and use its own MCP tools, and (2) a cache manager (Section 4.2), which caches and reuses previously discovered sources and navigation paths to make use of the structural dependencies among cells. To populate a target cell, the cache manager first queries neighboring page caches. If they do not contain the required information, it retrieves and generates a ranked list of candidate webpages from the path cache. The web explorer then executes the ATP-BFS traversal algorithm from these selected initial webpages in order.

Conceptually, SODIUMAGENT mirrors a processor-memory co-design: the web explorer functions as the core processing unit, executing the active exploration logic for each cell, while the cache manager serves as the storage unit, persistently maintaining previously discovered sources and structural paths to reduce redundant computation and improve coherence across cells. The synergistic interaction between these two components enables comprehensive and efficient collection and aggregation of data from the open web.

4.1 Web Explorer and the ATP-BFS Algorithm

We introduce the *augment-then-prune* breadth-first web exploration algorithm (ATP-BFS), an agentic web exploration algorithm (Algorithm 1) that serves as the backbone of the web explorer by conducting deep and comprehensive exploration on all visited webpages. Similar to classic BFS, ATP-BFS starts from a root webpage and iteratively expands a frontier queue in a layer-by-layer manner. For each visited page, outgoing links to unvisited webpages are added to the frontier until the system identifies and returns the target cell value.

As we illustrate in Figure 3, the web explorer executes ATP-BFS traversal with agentic interactions that interpret, interact with, and extract information from webpages during navigation.

We focus on two key innovations in the web explorer: (1) how it interacts with webpages to extract links and obtain answers (page explorer), and (2) how it determines the next layer of candidate links to explore through the *augment then prune* process (link processor).

Page Explorer. For each visited webpage u , the web explorer performs page-wise exploration and applies a decision function $\Phi(u)$ through the page explorer as a subagent:

$$\Phi(u) \in \{\text{Answer}(v), \text{Proceed}(L), \text{Stop}\},$$

where:

- $\text{ANSWER}(v)$ returns a value v for the target cell if the required information is present on u ;
- $\text{PROCEED}(L)$ returns a set of outgoing links L discovered on u for further exploration;
- STOP indicates that neither a valid answer nor useful links are found.

In practice, webpages differ substantially in how content is presented and can be categorized into three common classes: *static pages*, where all relevant content is fully revealed in the HTML source, *dynamically loaded pages*, where critical information may be hidden behind client-side rendering, JavaScript-driven UI components, or interactive elements such as tabs, accordions, pagination controls, and/or filters, and *online documents* (e.g., PDFs or images such as PNG/JPG/WebP).

When the web explorer encounters an online document URL, it invokes a file inspector that downloads the file for visual inspection. For image files, the inspector directly analyzes the image. For PDFs, it renders pages to images and examines them sequentially. After inspecting each image, the inspector decides whether the target cell value is already visible; if not, whether the answer is likely to appear on another page of the same file (and thus exploration should continue within the file), or whether further file exploration is unlikely to succeed, enabling early termination.

For webpages, the web explorer next classifies whether the page can be reliably treated as a static webpage or requires dynamic interactions. To do so, it compares two textual views extracted from the same webpage: a higher-fidelity observation obtained after rendering (which includes dynamically loaded or UI-revealed contents), and a lower-fidelity extract derived from static HTML. The page is treated as dynamic if the static extract is considered to be incomplete for the target cell. Specifically, a page is classified as dynamic if any of the following conditions hold: (i) the higher-fidelity observation contains concrete, query-relevant facts such as numbers, dates, names, or table rows that are missing from the static extract; (ii) the higher-fidelity observation reveals structural cues such as section headers, tabs, pagination indicators, or navigation items whose corresponding content is absent from the static extract; (iii) the static extract appears truncated, boilerplate, or substantially less informative than the rendered observation; or (iv) the static extract itself exposes UI controls (e.g., tabs, dropdowns, filters, or “load more” elements) without displaying their associated content.

If the page is classified as static, the page explorer directly extracts its visible text and outgoing links from the static HTML

component, and uses these signals together with a rendered screenshot to decide the return value among the three outcomes as we previously defined. If the page is classified as dynamic, the page explorer switches to an interactive exploration mode in which it incrementally reveals hidden contents.

Specifically, for dynamic webpages, the page explorer operates in a multi-turn manner. In each turn, it is provided with the webpage’s JavaScript accessibility structure and textual observations. It begins with an initial accessibility snapshot that includes common interactive components such as buttons and tabs. At each turn, the explorer performs one of the following 5 page-local actions:

- (1) `update_accessibility`: this operation allows the web explorer to dynamically extend its interaction interface by identifying specialized UI elements (e.g., custom-styled tabs or buttons) and registering them as clickable components. In effect, it enables the agent to design its own MCP tools by constructing page-specific interaction primitives.
- (2) `click`: this operation enables the agent to interact with one or more UI elements (e.g., expand/collapse, “show more”, or tab switches) to reveal additional on-page content.
- (3) `answer`: this operation terminates page-wise exploration when the required cell value becomes visible on the current page. It corresponds to the Answer outcome.
- (4) `extract_links`: this operation surfaces all visible hyperlinks for continued exploration when the answer is not found on the current page but the page may lead to relevant sources. It corresponds to the Proceed outcome.
- (5) `stop`: this operation terminates exploration when neither the answer nor useful outgoing links are present on the current page. It corresponds to the Stop outcome.

The web explorer is guided to fully apply (1) and (2) to reveal all available on-page contents before executing any of (3), (4), or (5).

During page exploration, the web explorer uses existing information as contextual signals to support its decision-making. Conversely, when it encounters clear and unambiguous evidence that contradicts earlier entries, it proactively revises the affected cell values to maintain consistency. This self-correction mechanism improves the robustness of SODIUMAGENT.

Link processor. If we were to follow the traditional BFS algorithm, the web explorer would (1) enqueue all outgoing links on each visited page, (2) explore only the links generated from the previous layers, and (3) traverse them strictly in the order they are added to the queue. Such a strategy is inefficient for web exploration for several reasons. First, a single webpage can contain a large number of outgoing links, causing the frontier size to grow rapidly and exponentially with depth. Second, the information required to answer a query, especially dated or archival content, may reside deep in the website hierarchy rather than within a few navigation steps from the root. Third, outgoing links on a webpage are not ordered by relevance to the query, which further delays reaching the target page and exacerbates the inefficiency caused by the large branching factor.

To address these challenges, we develop a link processor that operationalizes the *augment-then-prune* strategy in ATP-BFS. Given the extracted outgoing links together with their annotation texts,

the link processor produces a compact, relevance-oriented frontier for the next BFS layer. Specifically, it performs three stages: *Augment*, *Select*, and *Rank*.

- (1) *Augment (pattern-based URL proposal)*. The outgoing links on a webpage are often incomplete for the current query target (e.g., a page contains links for the year 2024, but the query requires year 2025). In the augment stage, we propose additional candidate URLs by inferring local URL patterns from the observed links and anchor texts. Specifically, the link processor analyzes the set of links listed on the webpage for recurring templates such as path segments (e.g., `/results/2024/`), slugs, identifiers, pagination markers, or query-string conventions. It then generates new URLs via minimal edits to existing on-page links, which preserves the site’s structure and avoids unrealistic hallucinations. A minimal edit changes at most one path segment (or applies a uniform multi-segment substitution such as `2024`→`2025` across all occurrences), and does not introduce new query parameters or additional path depth. In rare cases where the observed links imply multiple closely related domains (e.g., sibling subdomains `liheap-fy24-data-dashboard-hhs-acf.hub.arcgis.com` and `liheap-fy25-data-dashboard-hhs-acf.hub.arcgis.com`), the link processor performs a domain edit while keeping the full path unchanged.
- (2) *Select (joint pruning under a budget)*. After augmentation, we obtain a joint pool consisting of (i) the original outgoing links extracted from the current page and (ii) the newly proposed links from the augment stage. Exploring all of these links is impractical due to the large branching factor, so the select stage prunes this pool to a budget of at most K links (user-configurable). Selection is based on the textual descriptions available for each URL, including anchor text, surrounding snippet text, and URL tokens (e.g., year segments, team names, keywords). The link processor is guided to reserve a reasonable portion of the K budget for original links and the remainder for augmented links, preventing the frontier from collapsing to only augmented URLs or only on-page links.
- (3) *Rank (query and structure aware ordering)*. Finally, the selected links are ranked to determine the exploration order for the next BFS layer. Ranking is conditioned on (i) the user query, (ii) the target cell specification (the column name and row identifier), and (iii) any already collected evidence. The link processor prioritizes URLs whose textual descriptions suggest they directly contain the target value, as well as URLs that likely serve as intermediate navigation hubs toward relevant content.

The final output of the link processor after the three stages is a ranked list of up to K URLs that becomes the next-layer frontier for ATP-BFS.

4.2 Cache Manager

While the web explorer focuses on breadth-first exploration to ensure coverage, the cache manager complements it by exploiting

Algorithm 1: ATP-BFS Agentic Web Exploration

Input: Query q , target cell (r, c) , root URL u_0 , explore width K
Output: Cell value or \emptyset
Initialize frontier queue $Q \leftarrow [u_0]$;
Initialize visited set $V \leftarrow \emptyset$;
while $Q \neq \emptyset$ **do**
 $u \leftarrow \text{Dequeue}(Q)$;
 if $u \in V$ **then**
 \perp **continue**
 $V \leftarrow V \cup \{u\}$;
 $page \leftarrow \text{Fetch}(u)$;
 // Page-wise exploration
 $result \leftarrow \text{PageExplorer}(page, q, (r, c))$;
 if $result = \text{Answer}(v)$ **then**
 \perp **return** v
 if $result = \text{Proceed}(L)$ **then**
 // Link augment-then-prune
 $L_{next} \leftarrow \text{LinkProcessor}(L, q, (r, c), K)$;
 foreach $u' \in L_{next}$ **do**
 if $u' \notin V$ **then**
 \perp $\text{Enqueue}(Q, u')$;
 // else $result = \text{Stop}$
return \emptyset

depth-wise regularities in website structures, resembling a constrained depth-first search over previously successful navigation paths. The key insight is that neighboring cells within the same table, as well as neighboring webpages within the same domain, exhibit strong structural dependencies: their values are often located on the same pages or along highly similar navigation paths.

For each successfully filled cell, SODIUMAGENT caches both the final source page and the full exploration path from the initial root to that page. Caching the entire path is crucial because intermediate pages often encode structural cues (e.g., year indices, entity lists, or category hubs) that generalize across neighboring cells.

SODIUMAGENT’s cache is organized into two levels, both indexed by primary key and column. The *page-level cache* records mappings from the source webpages to the extracted values, enabling immediate reuse when multiple cell values are populated on the same page. The *path-level cache* stores full exploration paths, allowing the system to reuse deeper navigation patterns when structurally similar cells are encountered.

Before invoking the web explorer for a target cell, SODIUMAGENT first consults the cache manager. It checks whether the source page used by the upper or left neighboring cell already contains the required information. If not, SODIUMAGENT retrieves the cached exploration paths from these neighboring cells and performs a path search to identify promising next pages to visit. Specifically, given the two cached paths, SODIUMAGENT ranks a set of candidate URLs of size at most K by prioritizing URLs whose patterns suggest relevance to the target primary key and column, and those located near likely divergence points where navigation paths for different cells begin to differ. When appropriate, new candidate URLs are proposed through minimal edits to existing path URLs, such as substituting year or identifier segments, while preserving the observed site structure, following the same principle as the link processor.

The resulting ranked candidate URLs (or an empty set for the first cell with no cache), together with the base domain, are then used as root nodes (i.e., the u_0 in Algorithm 1) for the web explorer. This cache-guided initialization not only reduces redundant exploration but also extends SODIUMAGENT’s search depth.

Overall, the architecture of SODIUMAGENT enables more comprehensive and in-depth web exploration than existing web search systems by systematically diving into deep and archival pages when necessary, while simultaneously reducing exploration cost through effective reuse of cached pages and navigation paths.

5 EXPERIMENTS

We first describe the experimental setup in Section 5.1. We then present the system-level performance of all agents on SODIUMBENCH in Section 5.2, including a detailed analysis across the three dimensions (C_0 , \mathcal{E}_1 , and \mathcal{E}_2) for solving SODIUM. To better understand the mechanisms underlying SODIUMAGENT’s performance, we provide component analyses of the web explorer in Section 5.3 and the cache manager in Section 5.4. Finally, in Section 5.5, we conduct ablation studies to evaluate the standalone performance of the web explorer and the incremental benefit of the cache manager.

5.1 Experiment Setup

We evaluate the performance of SODIUMAGENT and 6 baseline agents on SODIUMBENCH.

5.1.1 Baselines. We select the following 6 state-of-the-art agents with advanced web search capabilities as baselines.

AG2 [2]. AG2 is a modular agent framework that coordinates multiple specialized agents (for example, a planner, researcher, and executor) to solve tasks via iterative reasoning and tool use, including web search and information gathering. It emphasizes flexible agent composition and structured collaboration to improve reliability on multi-step problems.

AutoGPT [56]. AutoGPT is an autonomous agent that decomposes a high-level goal into subgoals, repeatedly plans, searches the web, and executes actions to make progress with minimal human intervention. It maintains short-term and long-term context (for example, via memory or logs) to support iterative refinement over long horizons.

AutoGen [36]. AutoGen is a multi-agent conversation framework where agents communicate through structured messages to jointly plan, retrieve information, and complete tasks. It supports tool-using agents (such as web browsing, code execution, and function calls) and is designed to make complex workflows easier to orchestrate and evaluate.

OpenAI ResearchBot [44]. The OpenAI ResearchBot is a research-focused agent designed to perform multi-step online investigation, including querying the web, reading sources, synthesizing evidence, and producing grounded summaries.

Open Deep Research [43]. Open Deep Research is an open-source reproduction of a “deep research” [23] style agent that conducts iterative web exploration, note-taking, and synthesis across a wide range of sources to answer complex queries. It follows a structured

research workflow that alternates between planning, searching, reading, and summarizing, aiming for comprehensive coverage and traceable reasoning.

WebVoyager [25]. WebVoyager is an end-to-end web navigation agent that interacts with real websites by following links, clicking UI elements, and extracting information from pages to complete tasks. It couples high-level planning with low-level browser actions, enabling it to handle workflows that require multi-page navigation and on-page interaction.

All agents operate under a cell-wise extraction setting, where each task run corresponds to filling a single table cell. For each run, agents are provided with already filled values as contextual information. All agents are prompted using gpt-5-2025-08-07 [46]. We set the exploration width $K = 10$ for SODIUMAGENT.

5.1.2 Evaluation Metrics. We first define the cell matching criterion that underlies all value-based comparisons. We then introduce our primary metric, followed by metrics tailored to different dimensions. Here, R and C denote the number of rows and columns in each table, and i and j represent row and column indexes.

Cell Matching Criteria. For each predicted cell $\hat{y}_{i,j}$ and ground-truth cell $y_{i,j}$, we adopt two complementary matching strategies:

(1) **Exact Match.**

$$\mathbb{I}_{\text{match}}(i, j) = \begin{cases} 1 & \text{if } \hat{y}_{i,j} = y_{i,j}, \\ 0 & \text{otherwise.} \end{cases}$$

This metric is strict and fully reproducible.

(2) **LLM-as-a-Judge.** Exact match can be overly strict when formatting differs, but semantics are equivalent (e.g., “1%” vs. “0.01”, rounding differences, or alternative naming conventions). To account for such cases, we use GPT-4o [45] as a semantic judge that outputs a binary decision following data agent evaluation traditions [67]:

$$\mathbb{I}_{\text{llm}}(i, j) \in \{0, 1\}.$$

Primary Metric: Task-Level Accuracy. Our primary metric evaluates the success rate of structured data construction at the *task level*. We compute the task accuracy as the percentage of correctly filled cells in the materialized table:

$$\text{TaskAcc} = \frac{\sum_{i=1}^R \sum_{j=1}^C \mathbb{I}(i, j)}{RC}.$$

We then average across *all tasks in SODIUMBENCH*. This metric directly reflects the overall success rate of structured data construction, capturing the integrated effectiveness of C_0 , $\mathcal{E}1$, and $\mathcal{E}2$.

Dimension-Level Metrics. Beyond the primary task-level metric, we introduce dimension-level metrics to disentangle performance across C_0 , $\mathcal{E}1$, and $\mathcal{E}2$.

(C_0) In-Depth Information Discovery. To assess fine-grained retrieval performance, we compute CellAcc as the average accuracy across *all cells in SODIUMBENCH*. This metric reflects the overall proportion of correctly retrieved values and directly captures the system’s ability to perform in-depth information discovery (C_0).

($\mathcal{E}1$) Structural Correlation Identification. To evaluate whether systems correctly identify and exploit structural dependencies across table cells, we measure full row and full column accuracy.

Full Row Accuracy:

$$\text{RowAcc}_i = \mathbb{I} \left(\sum_{j=1}^C \mathbb{I}(i, j) = C \right).$$

Full Column Accuracy:

$$\text{RowAcc}_j = \mathbb{I} \left(\sum_{i=1}^R \mathbb{I}(i, j) = R \right).$$

The row accuracy is averaged across *all rows in SODIUMBENCH*, and the column accuracy is averaged across *all columns in SODIUMBENCH*. These metrics reflect whether the system can consistently apply structural patterns and reuse navigation strategies across related entries, directly measuring $\mathcal{E}1$.

($\mathcal{E}2$) Structured Database Construction. We define table-level accuracy as the percentage of fully correct tables:

$$\text{TableAcc} = \mathbb{I} \left(\sum_{i=1}^R \sum_{j=1}^C \mathbb{I}(i, j) = RC \right).$$

The table accuracy is averaged across *all tasks in SODIUMBENCH*. This metric evaluates whether the system successfully constructs a complete, coherent, and queryable database instance, directly reflecting $\mathcal{E}2$.

Metric Hierarchy. These metrics form a natural hierarchy:

$$\text{CellAcc} \geq \text{RowAcc}, \text{ColAcc} \geq \text{TableAcc}.$$

Conceptually, the hierarchy mirrors the progression from local value retrieval to global relational integrity: Cell accuracy measures retrieval correctness (C_0), structural accuracy measures consistency across related entries ($\mathcal{E}1$), and table accuracy measures complete and coherent database construction ($\mathcal{E}2$).

5.2 SODIUMAGENT achieves promising performance on SODIUM tasks

We report the overall performance of SODIUMAGENT in Table 2. We have the following analysis and conclusions:

SODIUMAGENT outperforms SOTA agents. SODIUMAGENT significantly outperforms all baselines across every SODIUM dimension and evaluation setting. Under LLM-as-a-judge evaluation, SODIUMAGENT achieves 91.1% task accuracy, nearly doubling AG2 and exceeding AutoGPT and AutoGen by more than 2.6 \times and 3.1 \times , respectively. Compared to OpenAI ResearchBot, WebVoyager, and Open Deep Research, the margin becomes even more substantial, reaching up to 73 \times over Open Deep Research. This dominance is consistent under exact match evaluation, where SODIUMAGENT achieves 69.5% task accuracy: under the stricter exact-match metric, SODIUMAGENT consistently achieves approximately 1.9 \times the performance of AG2 and over 2.4 \times that of AutoGen.

For cell accuracy (C_0), SODIUMAGENT reaches 91.8% under the LLM-as-a-judge metric, which is more than 2 \times AG2 and nearly 3 \times AutoGen. It further substantially surpasses AutoGPT and OpenAI ResearchBot, achieving over a 5 \times improvement compared to the

Table 2: Accuracy (\uparrow) of different agents on SODIUMBENCH in percentages (%). GPT-4o columns refer to LLM-as-a-judge evaluations, and Match columns refer to exact match evaluations. The optimal values for each metric are highlighted with both bold and underline formatting. We report the rank of each agent under each metric in italicized parentheses.

System	Cell Acc. (\mathcal{C}_0)		Row Acc. (\mathcal{E}_1)		Column Acc. (\mathcal{E}_1)		Table Acc. (\mathcal{E}_2)		Task Acc. ($\mathcal{C}_0, \mathcal{E}_1, \mathcal{E}_2$)	
	GPT-4o	Match	GPT-4o	Match	GPT-4o	Match	GPT-4o	Match	GPT-4o	Match
AG2	45.51 (2)	36.39 (2)	15.10 (2)	8.17 (2)	18.69 (2)	15.61 (2)	1.90 (3)	0.95 (2)	46.48 (2)	35.90 (2)
AutoGPT	35.74 (3)	29.32 (3)	10.89 (3)	5.22 (3)	10.26 (4)	7.44 (4)	1.90 (3)	0.95 (2)	34.73 (3)	28.42 (3)
AutoGen	31.32 (4)	26.57 (4)	7.84 (4)	3.33 (4)	13.71 (3)	11.21 (3)	2.86 (2)	0.95 (2)	29.54 (4)	24.70 (4)
OpenAI ResearchBot	16.71 (5)	5.21 (5)	1.56 (5)	0.43 (5)	3.84 (5)	0.61 (5)	0.00 (5)	0.00 (5)	16.61 (5)	4.85 (5)
WebVoyager	2.05 (6)	1.12 (6)	0.24 (6)	0.00 (6)	0.61 (6)	0.14 (6)	0.00 (5)	0.00 (5)	2.93 (6)	1.54 (6)
Open Deep Research	1.35 (7)	0.51 (7)	0.00 (7)	0.00 (6)	0.00 (7)	0.00 (7)	0.00 (5)	0.00 (5)	1.24 (7)	0.35 (7)
SODIUMAGENT	<u>91.76 (1)</u>	<u>67.24 (1)</u>	<u>75.75 (1)</u>	<u>39.68 (1)</u>	<u>78.30 (1)</u>	<u>52.30 (1)</u>	<u>52.38 (1)</u>	<u>23.81 (1)</u>	<u>91.07 (1)</u>	<u>69.48 (1)</u>
SODIUMAGENT’s Web Explorer	86.64	60.91	61.07	27.65	64.25	38.73	31.43	11.43	84.37	60.04

latter. This demonstrates that SODIUMAGENT is highly reliable at fine-grained information discovery.

For row and column accuracy (\mathcal{E}_1), the gap widens further. On row accuracy under the LLM-as-a-judge metric, SODIUMAGENT achieves 75.8%, which is $5.0\times$ AG2, $6.9\times$ AutoGPT, and nearly $10\times$ AutoGen. On column accuracy under the LLM-as-a-judge metric, SODIUMAGENT reaches 78.30%, over $4.2\times$ AG2 and more than $7.6\times$ AutoGPT. These results indicate that SODIUMAGENT effectively identifies and leverages structural correlations across cells.

For table accuracy (\mathcal{E}_2), which requires globally consistent reasoning across the entire table, SODIUMAGENT achieves 52.4% using the LLM-as-a-judge metric. This corresponds to a $27\times$ improvement over AG2 and an $18\times$ improvement over AutoGen. Under exact match, SODIUMAGENT achieves 23.8%, whereas all major baselines remain at or below 0.95%. This large gap demonstrates that most existing agents fail to maintain global consistency across rows and columns, while SODIUMAGENT successfully integrates \mathcal{C}_0 and \mathcal{E}_1 with coherent table-level reasoning (\mathcal{E}_2).

The large performance gaps at \mathcal{E}_1 and \mathcal{E}_2 dimensions underscore that structured multi-level reasoning, rather than isolated retrieval, is the key bottleneck in SODIUM tasks. Overall, these results demonstrate that existing web agents struggle with coordinated structural reasoning, whereas SODIUMAGENT successfully integrates in-depth information discovery (\mathcal{C}_0), structural dependency modeling (\mathcal{E}_1), and global table consistency (\mathcal{E}_2) into a unified and scalable system.

Consistent, robust, and complementary evaluation metrics enable reliable and discriminative comparisons on SODIUM tasks. Within the same accuracy level, the trends between exact match and LLM-as-a-judge are highly consistent in ranking, confirming the robustness of our conclusions. The LLM-based evaluation typically yields slightly higher scores because it tolerates semantically equivalent answers that differ in surface form.

More importantly, LLM-as-a-judge provides finer granularity when accuracy is low, enlarging performance gaps that exact match cannot distinguish. For example, both WebVoyager and Open Deep Research have 0% row accuracy under exact match. However, under LLM evaluation, WebVoyager achieves 0.24% while Open Deep Research remains at 0%, indicating that WebVoyager exhibits stronger

\mathcal{E}_1 than Open Deep Research. Similarly, for table accuracy, AutoGen achieves 2.86% under LLM evaluation, outperforming AG2 and AutoGPT (both 1.90%), while exact match reports identical scores (0.95%), masking this difference.

From the value perspective, across evaluation levels, we observe strong correlations between the three dimensions and overall task accuracy. The closeness between task and cell accuracy indicates that strong fine-grained retrieval (\mathcal{C}_0) forms a necessary foundation for end-to-end success. The importance of structural extensions (\mathcal{E}_1 and \mathcal{E}_2) becomes evident when moving from cell-level to row-, column-, and table-level metrics. Systems with moderate \mathcal{C}_0 but weak \mathcal{E}_1 and \mathcal{E}_2 quickly plateau as structural constraints become stricter. For example, the strongest baseline, AG2, experiences a substantial degradation: its row and column accuracy drop to less than 50% of its cell accuracy, and its table accuracy falls to below 5% of its cell accuracy. In contrast, SODIUMAGENT maintains consistently high performance from cell to table to task accuracy, indicating that the three SODIUM dimensions compose effectively rather than degrading at higher structural levels.

From the ranking perspective, the rankings of agents at cell and task accuracy levels are exactly the same. This one-to-one correspondence indicates that \mathcal{C}_0 serves as the backbone of end-to-end task success in SODIUM tasks, with improvements in cell-level accuracy translating directly into gains in overall task performance.

Stratification emerges at the row, column, and table accuracy levels, corresponding to \mathcal{E}_1 and \mathcal{E}_2 . At these higher structural levels, SODIUMAGENT is exceptionally dominant, substantially outperforming all baselines. AG2, AutoGen, and AutoGPT form a second tier of sophisticated agentic frameworks with explicit multi-step planning and tool-use capabilities, but their performance drops sharply when moving from cell-level correctness to row- and table-level consistency. OpenAI ResearchBot operates at a noticeably lower tier as a comparatively lightweight research-oriented agent, with more constrained planning and reasoning mechanisms. As a result, it struggles to maintain coherent multi-cell reasoning beyond basic retrieval. WebVoyager and Open Deep Research remain at the bottom tier; these systems are more specialized for browsing or exploratory research scenarios rather than structured data construction, and

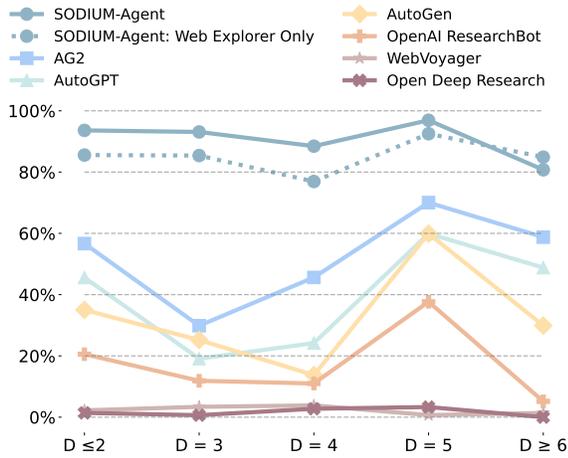


Figure 4: LLM-as-a-judge accuracy of different agents across different search depths D .

they exhibit almost no reliable multi-cell or table-level reasoning capability. This tiered pattern reveals an important insight: while many agents can occasionally retrieve correct individual values, none of them can systematically exploit structural dependencies across rows and columns or maintain global consistency across an entire table.

Taken together, these findings validate the design of our evaluation framework: by combining strict reproducibility, semantic robustness, and structural stratification, our evaluation metrics provide reliable, discriminative, and interpretable comparisons across agents on SODIUM tasks.

SODIUMAGENT achieves consistently superior performance across search depths. As we show in Figure 4, SODIUMAGENT maintains stable and consistently high accuracy across all search depths, substantially outperforming all baselines.

Interestingly, performance does not monotonically decrease with depth for all agents. Instead, we observe a non-linear pattern: accuracy drops from shallow pages ($D \leq 2$) to intermediate layers ($D = 3$), then improves and peaks around $D = 5$, before decreasing again at very deep levels ($D \geq 6$). This behavior reflects structural properties of real-world websites: agent performance drops from $D \leq 2$ to $D = 3$ due to increased exploration difficulty, as deeper navigation requires additional decisions, each introducing further opportunities for error. As search depth increases to $D = 3$ – 5 , navigation difficulty remains relatively stable. However, deeper pages tend to be more specialized and narrowly scoped. This reduces cross-topic interference and makes extraction more reliable once the correct page is reached, resulting in an increase in agent performance. However, at deeper levels ($D \geq 6$), exploration complexity becomes dominant again, and this long-horizon brittleness leads to the final decline in performance.

Table 3: The execution statistics of SODIUMAGENT’s web explorer across ATP-BFS levels ℓ with (as an embedded component of SODIUMAGENT) and without the cache manager (as a standalone agent).

		$\ell = 0$	$\ell = 1$	$\ell = 2$	$\ell \geq 3$
# Cells	Embedded	306	185	69	44
	Standalone	276	1006	524	343
Accuracy (%)	Embedded	91.2	91.9	92.8	93.2
	Standalone	81.2	86.2	93.9	81.3
Prune rate (%)	Embedded	–	63.0	96.6	93.5
	Standalone	–	78.8	96.9	93.6
New URL rate (%)	Embedded	–	35.8	29.3	25.7
	Standalone	–	48.4	35.8	30.8

5.3 SODIUMAGENT’s web explorer is efficient and robust

As we show in Table 3, SODIUMAGENT’s web explorer achieves consistently high accuracy across ATP-BFS traversal levels, with most cells resolved at shallow levels. Here, we use ℓ to refer to the traversal level to differentiate the search depth D in Section 5.2, with the latter referring to the intrinsic structural depth of the target cell values.

SODIUMAGENT resolves most cells with shallow exploration.

A large majority of cells are completed with $\ell \leq 1$ (306 at $\ell = 0$ and 185 at $\ell = 1$), while only 113 cells require deeper exploration ($\ell \geq 2$). This indicates that SODIUMAGENT is generally able to identify the correct region of a website and navigate to the target sources quickly, either through cache-assisted inference ($\ell = 0$) or with a single sorting iteration ($\ell = 1$). Combining the results in Table 3 and Table 4, we can see that out of the 500 cells that start from cached paths, in 306 of them (over 60%), the cache manager directly outputs the correct source as the initial point of entry for the web explorer. Across all cells, the average number of pages explored is 8.2 (min 1, max 172, median 3), suggesting that effective retrieval typically requires only a small number of page inspections.

Accuracy is stable across search depths and page exploration.

Accuracy remains remarkably stable across depths, ranging from 91.2% at $\ell = 0$ to 93.2% at $\ell \geq 3$. Importantly, we observe no meaningful relationship between the number of pages visited and correctness: the Pearson correlation [48] between page count and accuracy is -0.063 , and the Spearman correlation [58] is 0.054 , both well below 0.1. This indicates that deeper or broader exploration does not inherently degrade accuracy; instead, correctness depends on structural reasoning rather than search length.

Web explorer stabilizes the search space at deeper levels. The prune rate (i.e., the proportion of candidate webpages discarded at each depth step) rises sharply from $\ell = 1$ to $\ell = 2$, coinciding with an approximately $K = 10\times$ increase in the number of visited webpages. Beyond $\ell = 2$, both the number of visited pages per level and the prune rate stabilize, remaining consistently high. This indicates that ATP-BFS effectively compresses the expanded search

Table 4: Comparison of SODIUMAGENT components in the utility of cached results during cell filling. Accuracy is measured using LLM-as-a-judge evaluation, which provides finer-grained assessment, as analyzed in Section 5.2.

Component	Mode	# Cells	Acc. (%) \uparrow	Cost (\$) \downarrow
Web Explorer	Start from base	104	87.50	1.08
	Start from cached paths	500	92.60	0.91
Cache Manager	Left	885	93.67	0.23
	Up	660	89.24	0.16

space and prevents uncontrolled growth as exploration proceeds, maintaining stable and selective traversal even at deeper levels.

Meanwhile, the new URL rate gradually decreases with depth (35.8% \rightarrow 29.3% \rightarrow 25.7%). This suggests that URL synthesis is most beneficial in early exploration stages, when structural patterns are still being inferred. As depth increases and navigation becomes more grounded and stabilized in observed links, the need for speculative URL generation naturally diminishes.

Dynamic webpages dominate web exploration. Across all cells and stages, `inspect_dynamic` constitutes the vast majority (91.18%) of page inspections. SODIUMAGENT also typically *finishes* on dynamic pages: the last inspection call is dynamic for 1,901 cells versus 220 static and 28 online documents. Accuracy is slightly higher when the final inspection is dynamic (92.43%) than when it is static (90.91%), suggesting that dynamic endpoint pages are reliable for extracting the target values.

5.4 SODIUMAGENT’s cache manager improves accuracy and reduces costs

We report the cache utility results in Table 4. Across the total of 2,149 evaluated cells, we have the following observations.

Cached results are heavily used. Out of 2,149 cells, 1,545 cells (885 Left + 660 Up) are answered directly by the Cache Manager, accounting for 71.9% of all cells. An additional 500 cells (23.27%) use cached navigation paths before invoking the web explorer. Only 104 cells (4.8%) start exploration directly from the base URL without any cache assistance. Overall, 2045 out of 2,149 cells (95.2%) benefit from caching, demonstrating that cache reuse is the dominant execution pattern in SODIUMAGENT.

Cache improves accuracy. Direct cache usage (Left + Up) achieves the highest accuracy. The Left cache direction reaches 93.7%, while the Up direction achieves 89.2%. Both outperform starting from base (87.5%). Even when the system must fall back to web exploration, starting from cached paths improves accuracy from 87.5% to 92.6%, a gain of 5.1 percentage points. These results demonstrate that structural reuse strengthens both efficiency and robustness: the cache manager reduces redundant traversal while simultaneously increasing extraction accuracy.

Cache reduces cost. To quantify efficiency, we measure the API cost incurred per cell. Direct cache retrieval substantially lowers exploration cost. On average, the Left and Up cache directions incur only \$0.25 and \$0.18 per cell, respectively. In contrast, web exploration starting from the base page costs \$1.08 per cell, which is 19%

higher than exploration initialized from cached paths. These results demonstrate that structural reuse effectively reduces browsing overhead and overall token expenditure.

Cached path search effectively balances reuse and synthesis.

Across cells that use cached paths, SODIUMAGENT proposes an average of 4.44 candidate URLs per call. Notably, 60.4% of these URLs are newly synthesized rather than directly observed from the current page context. This indicates that the cache manager does not merely reuse previously discovered links, but actively generalizes structural patterns to generate plausible navigation paths. The relatively small proposal set keeps the branching factor low, while the high proportion of synthesized URLs allows SODIUMAGENT to actively adapt to unseen yet structurally similar cases.

Overall, these results show that the cache manager is responsible for handling nearly two-thirds of all cells directly, improving both accuracy and efficiency, while cached path reuse further enhances robustness when exploration is unavoidable.

5.5 \mathcal{E}_1 and \mathcal{E}_2 are critical for SODIUMAGENT

We ablate the cache manager and run the web explorer in isolation to assess its standalone performance and quantify the incremental contribution of caching. The standalone web explorer already outperforms all baseline agents by a substantial margin, and the cache manager consistently lowers the cost of SODIUMAGENT while further improving accuracy, reinforcing the conclusions drawn in Section 5.4. We detail our findings below.

The standalone web explorer substantially outperforms all baselines. As we show in Table 2, the web explorer alone significantly outperforms all baseline agents across every metric. In particular, it achieves 84.37% task-level accuracy under LLM-as-a-judge evaluation, compared to 46.48% for the strongest baseline, AG2, corresponding to a 1.82 \times improvement.

This advantage is consistent across structural levels. At the cell level, the standalone web explorer achieves over 1.9 \times the accuracy of AG2. At the row and column levels, the gains are even more pronounced: over 4 \times at the row level and over 3 \times at the column level. At the table level, where multi-cell consistency is critical, the web explorer achieves a 17 \times improvement over AG2.

As the web explorer constitutes the core retrieval component of SODIUMAGENT, these results reinforce our earlier claim that C_0 serves as a foundational capability. Once reliable deep retrieval is established, substantial performance gains over existing agentic systems can already be realized.

Cache manager and web explorer synergize to achieve strong structural integrity. Integrating the cache manager further improves performance across all levels. Task accuracy increases from 84.37% to 91.07%, a gain of 8% relative improvement.

The gains become more pronounced at stricter structural levels. At the cell level, accuracy increases by 5.12 percentage points, while row accuracy increases by 14.68 percentage points, column accuracy by 14.05 percentage points, and table accuracy by 20.95 percentage points. This demonstrates that while C_0 enables strong cell-wise retrieval, \mathcal{E}_1 and \mathcal{E}_2 , which leverage structural dependencies and

reuse discovered paths, are critical for maintaining global coherence across rows and entire tables.

In short, the web explorer provides depth and coverage, while the cache manager provides structural stability and consistency. Their combination allows SODIUMAGENT to exceed 90% task-level accuracy while dramatically improving table-level correctness.

Cache manager stabilizes SODIUMAGENT’s performance across different search depths. As we illustrate in Figure 4, the standalone web explorer exhibits noticeably larger performance fluctuations than SODIUMAGENT. The cache manager reduces this instability by reusing validated paths and previously discovered source formats. As a result, SODIUMAGENT consistently outperforms the standalone variant across most depths, with only minor differences at very large depths. This is as expected because as depth increases, the search space becomes more fragmented, with a higher branching factor; consequently, path reuse is more susceptible to minor structural divergence. However, this marginal degradation at extreme depths is insignificant relative to the pronounced overall gains delivered by structural reuse.

Cache manager significantly improves efficiency. Beyond accuracy, the cache manager substantially reduces exploration overhead. As we show in Table 3, the ATP-BFS traversal level distribution shifts markedly when the cache manager is removed.

In the embedded setting, 81.3% of cells are resolved within traversal levels 0 and 1, with only 18.7% requiring traversal level 2 or beyond. In contrast, without caching, the proportion of deeper explorations ($\ell \geq 2$) rises to 40.4%, more than doubling. The prune rates at all levels also increase in the standalone setting, indicating a larger fraction of discarded intermediate states and thus more ineffective browsing steps. Moreover, the new URL rate is consistently higher at all levels without caching, suggesting that the agent is more frequently navigating to previously unseen pages rather than reusing established structural patterns. This behavior reflects a lack of access to cached source formats and leads to more exploratory and redundant navigation.

This shift toward deeper exploration directly translates to higher cost. The average cost per cell is \$1.36 for the standalone web explorer and \$0.41 for SODIUMAGENT. Introducing the cache manager reduces cost by approximately 70%.

6 RELATED WORK

We review related work from the following three aspects.

LLM for Data Management. Data science workflows have traditionally been labor-intensive and iterative [12]. With the rapid development of LLMs, recent work has explored augmenting different stages of the data lifecycle using LLM agents and models. These efforts range from correcting and imputing missing values in tables [68], transforming unstructured data into structured tables [16, 27, 47, 54, 65], aggregating tables from multiple sources [21], to enabling downstream analytical workflows via NL2SQL [33, 35, 55] or more complex data science and machine learning pipelines [29, 40, 41, 71].

Despite this progress, large-scale data collection from the open web, often the critical first step in real-world data science pipelines [24, 59], remains relatively underexplored in terms of automation.

Existing approaches, whether based on traditional web scraping techniques [70] or LLM agents [28], typically assume that the target dataset is already well-organized and structurally complete on the web. In practice, however, relevant information is often scattered across heterogeneous webpages and requires integration and normalization, similar to constructing structured repositories such as data lakes. Addressing this gap is precisely the focus of SODIUM.

Open domain search tools. Search tools substantially enhance the performance of LLM agents by providing access to open-domain knowledge [2, 23, 36, 43, 56]. Modern web search tools integrated with LLMs also demonstrate promising capabilities [8–10].

Nevertheless, existing approaches exhibit two critical limitations when deep, domain-specific information is required. First, they are primarily optimized for general-purpose search. For example, the search engine [5] used to improve SearchR1 [30] targets broad, generic queries rather than structured, domain-specific data extraction tasks. Second, they lack sufficient exploration depth. WebVoyager [25] performs relatively shallow website navigation, and other search-based systems largely rely on external search engines without explicitly optimizing multi-step, in-site exploration strategies. Existing web scrapers are effective for static websites [1, 3, 6, 7]; however, real-world websites often require interactive behaviors, such as expanding tabs or triggering dynamic content loading, to fully access relevant information. As a result, these approaches are insufficient for SODIUM tasks, where answering a single query requires deep, structured traversal within specialized websites.

Information retrieval and RAG systems. Information retrieval and RAG systems have demonstrated strong effectiveness across diverse data modalities, including structured data [20], unstructured text [22, 34, 66], and graph data [26]. However, these approaches typically decouple retrieval from the structural organization of the target outputs, treating retrieval as an independent pre-processing step. Such separation is suboptimal for SODIUM tasks, where retrieval decisions and structural reasoning must be tightly integrated.

Moreover, even methods that study dynamic or multi-frame data settings [17] assume that all relevant data has already been pre-extracted and stored locally. That is, they operate over a static, predefined corpus and do not require active, multi-step retrieval from live webpages.

7 CONCLUSION

In this paper, we formalize the problem of structuring open-domain unstructured data into materialized databases as the SODIUM task. To quantitatively evaluate SODIUM in real-world settings, we collect SODIUMBENCH of 105 queries from 6 domains. To address this challenge, we develop SODIUMAGENT, an agentic system composed of a web explorer and a cache manager. We design a novel ATP-BFS algorithm for SODIUM tasks as a workflow for the web explorer to achieve deep, schema-driven exploration. The cache manager uses structural regularities across table cells to reuse validated navigation paths, enforce cross-cell consistency, and improve efficiency. We evaluate SODIUMAGENT on SODIUMBENCH and show that it achieves 91.1% task-level accuracy, significantly outperforming state-of-the-art baselines. These results demonstrate that treating open domains as latent, materializable databases is both feasible and essential for supporting scalable, real-world analytical workflows.

REFERENCES

- [1] 2024. Jina Reader API. <https://jina.ai/reader/>. An API that converts URLs into structured, LLM-friendly text.
- [2] 2026. AG2 Documentation. <https://docs.ag2.ai/>.
- [3] 2026. Crawllee: Scalable Web Scraping and Crawling Library. <https://crawllee.dev/>.
- [4] 2026. ExploraDati (IstatData): Data Browser for Istat Aggregated Statistics. <https://esploradati.istat.it/>.
- [5] 2026. FastAPI Documentation. <https://fastapi.tiangolo.com/>.
- [6] 2026. Go-Colly: Elegant Scraping Framework for Go. <https://go-colly.org/docs/>.
- [7] 2026. Scrapy: A Fast High-Level Web Crawling & Web Scraping Framework. <https://www.scrapy.org/>.
- [8] 2026. Web Search Tool ('google_web_search') — Gemini CLI Documentation. <https://www.geminicli.com/docs/tools/web-search/>.
- [9] 2026. Web Search Tool — Claude API Documentation. <https://platform.claude.com/docs/en/agents-and-tools/tool-use/web-search-tool>.
- [10] 2026. Web Search Tool — OpenAI API Guides. <https://developers.openai.com/api/docs/guides/tools-web-search>.
- [11] Sahar Ahmed, Georges Reniers, Bruno Masquelier, David A. Sánchez-Páez, Julio Romero-Prieto, and Tom Pullum. 2024. Sample selection bias in adult mortality estimates from mobile phone surveys: Evidence from 25 low- and middle-income countries. *Demographic Research* 51, 37 (None 2024), 1167–1182. <https://doi.org/10.4054/DemRes.2024.51.37>
- [12] Sara Alspaugh, Nava Zokaei, Andrea Liu, Cindy Jin, and Marti A. Hearst. 2019. Futzing and Moseying: Interviews with Professional Data Analysts on Exploration Practices. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 22–31. <https://doi.org/10.1109/TVCG.2018.2865040>
- [13] Phillip M Bellinger, Timothy Newans, Mitchell Whalen, and Clare L. Minahan. 2021. Quantifying the Activity Profile of Female Beach Volleyball Tournament Match-Play. *Journal of sports science & medicine* 20 1 (2021), 142–148. <https://api.semanticscholar.org/CorpusID:231949658>
- [14] Caroline Berghammer, Torkild Hovde Lyngstad, Anna Matysiak, and Francesca Rinesi. 2024. Is single parenthood increasingly an experience of less-educated mothers? A European comparison over five decades. *Demographic Research* 51, 34 (2024), 1059–1094. <https://doi.org/10.4054/DemRes.2024.51.34> arXiv:<https://www.demographic-research.org/volumes/vol51/34/51-34.pdf>
- [15] Filipe Oliveira Bicudo, Lucas Savassi Figueiredo, Amanda Franco da Silva, Hugo Sarmento, Jocelyn Solomons, Filipe Manuel Clemente, and Henrique de Oliveira Castro. 2025. Comparison of Ball-in-Play Running Demands Across Game Phases and The Relationship between Physical and Technical Variables: An Analysis of The 2024 Female Super Sevens Tournament. *Journal of Sports Science & Medicine* 24, 4 (2025), 713.
- [16] Asim Biswal, Liana Patel, Siddharth Jha, Amog Kamsetty, Shu Liu, Joseph E. Gonzalez, Carlos Guestrin, and Matei Zaharia. 2024. Text2SQL is Not Enough: Unifying AI and Databases with TAG. arXiv:2408.14717 [cs.DB] <https://arxiv.org/abs/2408.14717>
- [17] Jianguo Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. 2023. Continual Learning for Generative Retrieval over Dynamic Corpora. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*. ACM, 306–315. <https://doi.org/10.1145/3583780.3614821>
- [18] Mário Costa, Nuno Garrido, Daniel Marinho, and Catarina Santos. 2021. How Much the Swimming Performance Leading to Tokyo 2020 Olympic Games Was Impaired Due to the Covid-19 Lockdown? *Journal of Sports Science and Medicine* 20 (09 2021), 714–720. <https://doi.org/10.52082/jssm.2021.714>
- [19] Joseph Coyne, Aaron Coutts, Robert Newton, and G Gregory Haff. 2021. Training load, heart rate variability, direct current potential and elite long jump performance prior and during the 2016 Olympic Games. *Journal of Sports Science & Medicine* 20, 3 (2021), 482.
- [20] Liancheng Fang, Aiwei Liu, Hengrui Zhang, Henry Peng Zou, Weizhi Zhang, and Philip S. Yu. 2025. TABGEN-ICL: Residual-Aware In-Context Example Selection for Tabular Data Generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Vienna, Austria, 20027–20041. <https://doi.org/10.18653/v1/2025.findings-acl.1027>
- [21] Juliana Freire, Grace Fan, Benjamin Feuer, Christos Koutras, Yurong Liu, Eduardo Peña, Aécio S. R. Santos, Cláudio Silva, and Eden Wu. 2025. Large Language Models for Data Discovery and Integration: Challenges and Opportunities. *IEEE Data Eng. Bull.* 49 (2025), 3–31. <https://api.semanticscholar.org/CorpusID:277195386>
- [22] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL] <https://arxiv.org/abs/2312.10997>
- [23] Google. 2024. Gemini Deep Research. <https://gemini.google/overview/deep-research/>.
- [24] Orit Hazzan and Koby Mike. 2023. *The Data Science Workflow*. Springer International Publishing, Cham, 151–163. https://doi.org/10.1007/978-3-031-24758-3_10
- [25] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. arXiv preprint arXiv:2401.13919 (2024).
- [26] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. In *Advances in Neural Information Processing Systems*, A. Gelman, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 132876–132907. <https://doi.org/10.52202/079017-4224>
- [27] Chuxuan Hu, Austin Peters, and Daniel Kang. 2024. LEAP: LLM-Powered End-to-End Automatic Library for Processing Social Science Queries on Unstructured Data. *Proceedings of the VLDB Endowment* 18, 2 (Oct. 2024), 253–264. <https://doi.org/10.14778/3705829.3705843>
- [28] Chuxuan Hu, Maxwell Yang, James Weiland, Yeji Lim, Suhas Palawala, and Daniel Kang. 2025. DRAMA: Unifying Data Retrieval and Analysis for Open-Domain Analytic Queries. *Proceedings of the ACM on Management of Data* 3, 6 (Dec. 2025), 1–28. <https://doi.org/10.1145/3769781>
- [29] Chuxuan Hu, Liyun Zhang, Yeji Lim, Aum Wadhvani, Austin Peters, and Daniel Kang. 2025. REPRO-Bench: Can Agentic AI Systems Assess the Reproducibility of Social Science Research? arXiv:2507.18901 [cs.CL] <https://arxiv.org/abs/2507.18901>
- [30] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. arXiv:2503.09516 [cs.CL] <https://arxiv.org/abs/2503.09516>
- [31] Maxim Kan. 2024. Religion and contraceptive use in Kazakhstan: A study of mediating mechanisms. *Demographic Research* 50, 21 (2024), 547–582. <https://doi.org/10.4054/DemRes.2024.50.21> arXiv:<https://www.demographic-research.org/volumes/vol50/21/50-21.pdf>
- [32] Eugenie Lai, Gerardo Vitagliano, Ziyu Zhang, Om Chabra, Sivaprasad Sudhir, Anna Zeng, Anton A. Zabreyko, Chenning Li, Ferdi Kossmann, Jialin Ding, Jun Chen, Markos Markakis, Matthew Russo, Weiyang Wang, Ziniu Wu, Michael J. Cafarella, Lei Cao, Samuel Madden, and Tim Kraska. 2025. KramaBench: A Benchmark for AI Systems on Data-to-Insight Pipelines over Data Lakes. arXiv:2506.06541 [cs.DB] <https://arxiv.org/abs/2506.06541>
- [33] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. arXiv preprint arXiv:2411.07763 (2024).
- [34] 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. arXiv:2005.11401 [cs.CL] <https://arxiv.org/abs/2005.11401>
- [35] Jinyang Li, Binyuan Hui, Ge Qu, Jiayi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems* 36 (2024).
- [36] Microsoft. 2024. AutoGen: A Programming Framework for Agentic AI. <https://github.com/microsoft/autogen>.
- [37] Ryouhei Mogi, James M. Raymo, Miho Iwasawa, and Shohei Yoda. 2023. An alternative version of the second demographic transition? Changing pathways to first marriage in Japan. *Demographic Research* 49, 16 (2023), 423–464. <https://doi.org/10.4054/DemRes.2023.49.16> arXiv:<https://www.demographic-research.org/volumes/vol49/16/49-16.pdf>
- [38] Margherita Moretti, Alessandra De Rose, and Elisa Cisotto. 2024. Uncovering disability-free grandparenthood in Italy between 1998 and 2016 using gender-specific decomposition. *Demographic Research* 50, 42 (2024), 1247–1264. <https://doi.org/10.4054/DemRes.2024.50.42> arXiv:<https://www.demographic-research.org/volumes/vol50/42/50-42.pdf>
- [39] Jerônimo Muniz, Bernardo Lanza Queiroz, and Aliya Saperstein. 2024. Racial classification as a multistate process. *Demographic Research* 50, 17 (2024), 457–472. <https://doi.org/10.4054/DemRes.2024.50.17> arXiv:<https://www.demographic-research.org/volumes/vol50/17/50-17.pdf>
- [40] Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, and Tomas Pfister. 2025. DS-STAR: Data Science Agent via Iterative Planning and Verification. arXiv:2509.21825 [cs.AI] <https://arxiv.org/abs/2509.21825>
- [41] Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Jinwoo Shin, Sercan Ö. Arik, and Tomas Pfister. 2025. MLE-STAR: Machine Learning Engineering Agent via Search and Targeted Refinement. arXiv:2506.15692 [cs.LG] <https://arxiv.org/abs/2506.15692>
- [42] Cameron Nosrat, Adrian Vallejo, Kwaku Djan, Youssef Sibih, Brian Feeley, and Nirav Pandya. 2025. National Basketball Association Players' Return to Play and Performance After Operative Treatment of Meniscal Tears. *Journal of Sports Science and Medicine* 24 (06 2025), 363–369. <https://doi.org/10.52082/jssm.2025.363>

- [43] Open Deep Research. 2024. Open Deep Research. <https://opendeepresearch.vercel.app/>.
- [44] OpenAI. 2024. OpenAI Agents Python: Research Bot Example. https://github.com/openai/openai-agents-python/tree/main/examples/research_bot.
- [45] OpenAI. 2025. GPT-4o Model. <https://developers.openai.com/api/docs/models/gpt-4o>.
- [46] OpenAI. 2025. GPT-5 Model. <https://developers.openai.com/api/docs/models/gpt-5>.
- [47] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators: A Declarative Model for Rich, AI-based Data Processing. arXiv:2407.11418 [cs.DB] <https://arxiv.org/abs/2407.11418>
- [48] Karl Pearson. 1895. Notes on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London* 58 (June 1895), 240–242.
- [49] Liat Raz-Yurovich and Barbara S. Okun. 2024. Are highly educated partners really more gender egalitarian? A couple-level analysis of social class differentials in attitudes and behaviors. *Demographic Research* 50, 34 (2024), 1005–1038. <https://doi.org/10.4054/DemRes.2024.50.34> arXiv:<https://www.demographic-research.org/volumes/vol50/34/50-34.pdf>
- [50] Catarina Santos, Ricardo Fernandes, Daniel Marinho, and Mário Costa. 2023. From Entry to Finals: Progression and Variability of Swimming Performance at the 2022 FINA World Championships. *Journal of Sports Science and Medicine* 22 (09 2023), 417–424. <https://doi.org/10.52082/jssm.2023.417>
- [51] Catarina C Santos, Ricardo J Fernandes, Daniel A Marinho, and Mário J Costa. 2023. From entry to finals: progression and variability of swimming performance at the 2022 FINA world championships. *Journal of Sports Science & Medicine* 22, 3 (2023), 417.
- [52] Thomas Savidge. 2025. Enabling Bad Behavior: The Federal Reserve’s Municipal Liquidity Facility in Retrospect. <https://aier.org/article/enabling-bad-behavior-the-federal-reserves-municipal-liquidity-facility-in-retrospect/>.
- [53] Thomas Savidge. 2025. The Work vs Welfare Tradeoff Revisited. <https://aier.org/article/the-work-vs-welfare-tradeoff-revisited/>.
- [54] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. arXiv:2410.12189 [cs.DB] <https://arxiv.org/abs/2410.12189>
- [55] Vladislav Shkapenyuk, Divesh Srivastava, Theodore Johnson, and Parisa Ghane. 2025. Automatic Metadata Extraction for Text-to-SQL. arXiv:2505.19988 [cs.DB] <https://arxiv.org/abs/2505.19988>
- [56] Significant-Gravitas. 2023. AutoGPT: OpenAI GPT-Powered Autonomous Agent. <https://github.com/Significant-Gravitas/AutoGPT>.
- [57] Jason Sorens. 2025. Have Mount Laurel Obligations Made New Jersey Housing More Affordable? A Synthetic Control Analysis of Housing Supply and Cost. <https://aier.org/article/have-mount-laurel-obligations-made-new-jersey-housing-more-affordable-a-synthetic-control-analysis-of-housing-supply-and-cost/>.
- [58] C. Spearman. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology* 15, 1 (1904), 72–101. <http://www.jstor.org/stable/1412159>
- [59] Victoria Stodden. 2020. The data science life cycle: a disciplined approach to advancing data science as a science. *Commun. ACM* 63, 7 (June 2020), 58–66. <https://doi.org/10.1145/3360646>
- [60] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663 [cs.IR] <https://arxiv.org/abs/2104.08663>
- [61] Emily Treleaven and Emma Banchoff. 2024. Children under 5 in polygynous households in sub-Saharan Africa, 2000 to 2020. *Demographic Research* 51, 32 (2024), 999–1016. <https://doi.org/10.4054/DemRes.2024.51.32> arXiv:<https://www.demographic-research.org/volumes/vol51/32/51-32.pdf>
- [62] Amy Tsui, Philip Anglewicz, Dana Sarnak, Saifuddin Ahmed, Fredrick Makumbi, Georges Guiella, Rosine Mosso, and Peter Gichangi. 2024. Predictive utility of key family planning indicators on dynamic contraceptive outcomes: Results from longitudinal surveys in Burkina Faso, Kenya, Uganda, and Côte d’Ivoire. *Demographic Research* 50, 45 (2024), 1301–1352. <https://doi.org/10.4054/DemRes.2024.50.45> arXiv:<https://www.demographic-research.org/volumes/vol50/45/50-45.pdf>
- [63] Willem R. J. Vermeulen, Aart C. Liefbroer, Niels Kooiman, and Mioara Zoutewelle-Terovan. 2023. Religion and union dissolution: Effects of couple and municipal religiosity on divorce and separation. *Demographic Research* 49, 20 (2023), 513–542. <https://doi.org/10.4054/DemRes.2023.49.20> arXiv:<https://www.demographic-research.org/volumes/vol49/20/49-20.pdf>
- [64] Zang Wanli, Mingqing Fang, Zhang Xianzuo, Ningkun Xiao, Su Wang, and Liang Mu. 2023. Exploring the Epidemiology of Injuries in Athletes of the Olympic Winter Games: A Systematic Review and Meta-Analysis. *Journal of Sports Science and Medicine* 2023 (12 2023), 748–759. <https://doi.org/10.52082/jssm.2023.748>
- [65] Lindsey Linxi Wei, Shreya Shankar, Sepanta Zeighami, Yeounoh Chung, Fatma Ozcan, and Aditya G. Parameswaran. 2026. Multi-Objective Agentic Rewrites for Unstructured Data Processing. arXiv:2512.02289 [cs.DB] <https://arxiv.org/abs/2512.02289>
- [66] Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fanguyan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. 2025. RAG-Gym: Systematic Optimization of Language Agents for Retrieval-Augmented Generation. arXiv:2502.13957 [cs.CL] <https://arxiv.org/abs/2502.13957>
- [67] Bonnie Xu, Aravind Suresh, and Emma Tang. 2026. Inside OpenAI’s in-house data agent. <https://openai.com/index/inside-our-in-house-data-agent/>.
- [68] Chenyu Yang, Yuyu Luo, Chuanxuan Cui, Ju Fan, Chengliang Chai, and Nan Tang. 2025. Data Imputation with Limited Data Redundancy Using Data Lakes. *Proc. VLDB Endow.* 18, 10 (June 2025), 3354–3367. <https://doi.org/10.14778/3748191.3748200>
- [69] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. 2024. CRAG – Comprehensive RAG Benchmark. arXiv preprint arXiv:2406.04744 (2024). <https://arxiv.org/abs/2406.04744>
- [70] Haoxiang Zhang, Aécio Santos, and Juliana Freire. 2021. DSSD: Domain-Specific Dataset Discovery on the Web. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (Virtual Event, Queensland, Australia) (CIKM ’21)*. Association for Computing Machinery, New York, NY, USA, 2527–2536. <https://doi.org/10.1145/3459637.3482427>
- [71] Jiani Zhang, Hengrui Zhang, Rishav Chakravarti, Yiqun Hu, Patrick Ng, Asterios Katsifodimos, Huzefa Rangwala, George Karypis, and Alon Halevy. 2025. CoddLLM: Empowering Large Language Models for Data Analytics. arXiv:2502.00329 [cs.DB] <https://arxiv.org/abs/2502.00329>