DRAMA: Unifying Data Retrieval and Analysis for Open-Domain Analytic Queries

CHUXUAN HU, University of Illinois Urbana-Champaign, USA
MAXWELL YANG, University of Illinois Urbana-Champaign, USA
JAMES WEILAND, University of Illinois Urbana-Champaign, USA
YEJI LIM, University of Illinois Urbana-Champaign, USA
SUHAS PALAWALA, University of Illinois Urbana-Champaign, USA
DANIEL KANG, University of Illinois Urbana-Champaign, USA

Manually conducting real-world data analyses is labor-intensive and inefficient. Despite numerous attempts to automate data science workflows, none of the existing paradigms or systems fully demonstrate all three key capabilities required to support them effectively: (1) open-domain data collection, (2) structured data transformation, and (3) analytic reasoning.

To overcome these limitations, we propose DRAMA, an end-to-end paradigm that answers users' analytic queries in natural language on large-scale open-domain data. DRAMA unifies data collection, transformation, and analysis as a single pipeline. To quantitatively evaluate system performance on tasks representative of DRAMA, we construct a benchmark, DRAMABENCH, consisting of two categories of tasks: claim verification and question answering, each comprising 100 instances. These tasks are derived from real-world applications that have gained significant public attention and require the retrieval and analysis of open-domain data. We develop DRAMABOT, a multi-agent system designed following DRAMA. It comprises a data retriever that collects and transforms data by coordinating the execution of sub-agents, and a data analyzer that performs structured reasoning over the retrieved data. We evaluate DRAMABOT on DRAMABENCH together with five state-of-the-art baseline agents. DRAMABOT achieves 86.5% task accuracy at a cost of $0.05, outperforming all baselines with up to 6.9× the accuracy and less than 1/6 of the cost. DRAMA is publicly available at https://github.com/uiuc-kang-lab/drama.

CCS Concepts: • **Information systems → Data management systems**.

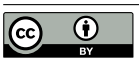
Additional Key Words and Phrases: Agentic Systems, Multi-Agent Systems, Dynamic Databases, AI for Data Science, AI for Databases, Natural Language Interfaces to Data, Data Integration, Information Retrieval

ACM Reference Format:
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. *Proc. ACM Manag. Data* 3, 6 (SIGMOD), Article 316 (December 2025), 28 pages. https://doi.org/10.1145/3769781

1 Introduction

Data science workflows can generally be divided into two phases: data retrieval and data analysis [22, 60] (Figure 1). In real-world scenarios, analysts must navigate data that is both large in volume and rapidly evolving. Successfully completing tasks grounded in such data with high accuracy

Authors' Contact Information: Chuxuan Hu, University of Illinois Urbana-Champaign, Urbana, Illinois, USA, chuxuan3@illinois.edu; Maxwell Yang, University of Illinois Urbana-Champaign, Urbana, Illinois, USA; James Weiland, University of Illinois Urbana-Champaign, Urbana, Illinois, USA; Yeji Lim, University of Illinois Urbana-Champaign, Urbana, Illinois, USA; Suhas Palawala, University of Illinois Urbana-Champaign, Urbana, Illinois, USA; Daniel Kang, University of Illinois Urbana-Champaign, Urbana, Illinois, USA, ddkang@illinois.edu.



ACM 2836-6573/2025/12-ART316
https://doi.org/10.1145/3769781

requires the integration of three key capabilities: **(C1)** open-domain data collection, **(C2)** structured data transformation, and **(C3)** analytic reasoning.

Consider a data analyst manually answering the question: "What is the national park with the highest visitor spending in 2023 in US?" [66] (Q_0). To obtain the newly released 2023 data, the analyst needs to first locate and download the relevant and authoritative data from an official government website [44] **(C1)**. The collected data is a 68-page PDF report on National Park Visitor Spending Effects [43] (i.e., unstructured data). Table 5, spanning pages 27 to 39, contains total visitor spending information for all 433 National Park System (NPS) units, listed in alphabetical order by name. To answer Q_0, the analyst further needs to manually extract and digitize this information by, for example, entering the data into a CSV file **(C2)**. Finally, the analyst must use analytic reasoning over structured data **(C3)**. This involves filtering the 63 national parks, which are a subset of NPS units and can only be identified by the suffix "NP" in the park unit name (e.g., "Acadia NP"). The analyst must therefore recognize that this information is embedded in the name itself rather than stored in a separate column and correctly interpret the semantic role of the suffixes. The analyst must also determine the correct field among seven numeric columns and then identify the maximum value. This process results in a query such as:

```
SELECT park_unit, total_visitor_spending FROM table
WHERE park_unit LIKE '%NP'
ORDER BY total_visitor_spending DESC LIMIT 1
```

Traditional data science workflows are tedious and time-consuming [1], highlighting the critical need for automation. However, existing systems face two key limitations that hinder full automation:

First, from the perspective of end-to-end completeness, as we outline in Figure 1, none of the existing paradigms or systems are capable of simultaneously demonstrating all of **(C1)**–**(C3)**. Specifically, existing open-domain information extraction (IE) systems [11, 12, 29] and web search tools [23, 49, 51, 61, 80] actively collect data from the web with no guaranteed accuracy through computations grounded in data. Existing retrieval-augmented generation (RAG)-based systems [2, 7, 62], even those incorporating dynamic knowledge base construction [30, 40, 72], assume that all relevant documents have already been collected from the open domains. They do not actively collect up-to-date or task-specific data, and instead operate on a fixed set. Both open-domain retrieval systems and RAG-based approaches lack support for analytic reasoning over structured data, as their outputs rely on surface-level text generation rather than computations grounded in the underlying data. Existing data analytic tools [2, 5, 10, 17, 24, 25, 37, 52, 56, 81], on the other hand, support accurate computations over structured data, but assume that the data is readily collected and transformed. In fact, except for TAG [2], which unifies RAG **(C2)** and query generation **(C3)**, all other existing paradigms and systems focus on only one of the three key capabilities.

Second, although existing paradigms and systems target subsets of **(C1)**–**(C3)**, their corresponding capabilities rely on strong and often unrealistic assumptions, and deteriorate when placed within end-to-end settings. Thus, simply aggregating systems with different capabilities is insufficient to automate the full pipeline. Specifically, open-domain IE systems and web search tools operate independently of downstream transformation (**C2**) and analysis (**C3**), limiting them to simple fact lookups and preventing them from supporting large-scale data collection, thus only partially realizing **(C1)**. RAG-based systems presume that data collected from open domains is already clean, structured, and uniform, making them unable to handle data heterogeneity. Their separation from downstream analytic requirements also prevents them from transforming the data to meet analytic needs, such as renaming columns with meaningful labels or aligning schemas. **(C2)** is thus not fully realized. The isolation of data analytic tools from upstream data retrieval

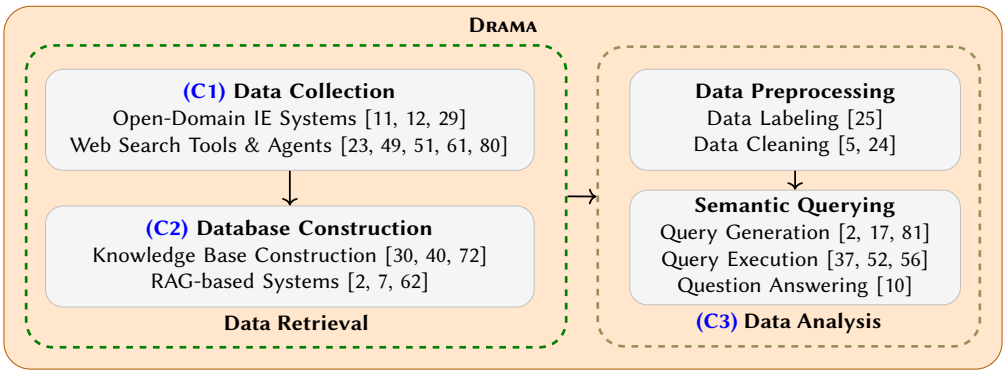


Fig. 1. DRAMA integrates the full data science pipeline, where (C1)–(C3) correspond to the essential capabilities underpinning each stage. Here, "Data Retrieval" refers to the collection and structuring of raw data from open domains.

makes them less flexible to the schema variations, including interpreting contextual cues such as embedded keywords in column values in the previous example, limiting their effectiveness in **(C3)**.

To address the limitations of existing systems within the scope of tasks that require both (1) large-scale, real-world data collection from open domains and (2) complex analysis and structured reasoning over the collected data to support decision-making, we introduce a new end-to-end paradigm DRAMA, the **D**ata **R**etrieval and **A**nalytical **MA**nagement pipeline (Section 2). This paradigm answers a user's analytic query Q in natural language with an answer A by integrating data collection, transformation, and analysis. Specifically, DRAMA consists of three key stages, as illustrated in Figure 2. First, the data collection stage, `collect`, collects the necessary raw data D from open domains based on the query Q. The collected data can appear in various formats, ranging from Excel files to PDF reports, depending on the nature of the data source. Second, the data transformation stage, `transform`, constructs a structured database T based on both D and Q. This stage extracts information from raw data and organize it for downstream analytic reasoning. Third, the data analysis stage, `analyze`, answers Q over T with A as the final result. DRAMA is unified by the following three simple equations and applies broadly to real-world scenarios.

$$\text{Data Collection:} \quad \texttt{collect}(Q) \rightarrow D \tag{1}$$

$$\text{Data Transformation:} \quad \texttt{transform}(Q, D) \rightarrow T \tag{2}$$

$$\text{Data Analysis:} \quad \texttt{analyze}(Q, T) \rightarrow A \tag{3}$$

To quantitatively demonstrate system performance on real-world tasks that capture the essential challenges of DRAMA, we introduce DRAMABENCH in Section 3. DRAMABENCH consists of two categories of tasks: (1) claim verification and (2) question answering, each comprising 100 task instances. Unlike existing benchmarks, DRAMABENCH is designed to reflect the challenges of performing both large-scale, open-domain data collection and complex analytical reasoning over the collected data, which are not reflected in existing benchmarks. Specifically, existing open-domain claim verification and question answering benchmarks (1) focus on simple text lookups rather than large-scale, real-world data collection, and (2) evaluate system performance based on textual similarity rather than the analytic reasoning required for decision-making [23, 38]. Existing data-grounded verification and question answering benchmarks assume that a static and fully collected data is readily available [33, 36], bypassing the challenges of dynamic and large-scale data collection. As a result, no existing benchmarks adequately capture the scope or application scenarios

that Drama is designed to address. To fill this gap, DramaBench provides tasks drawn from content dated on or after January 1st, 2024, which is later than the knowledge cutoffs of state-of-the-art large language models (LLMs). This ensures that completing them requires collecting data from open domains. The claim verification tasks consist of real-world false claims posted on social media that have been publicly corrected using verifiable, publicly available data. The question answering tasks involve analytic factual questions addressing socially and economically impactful topics, with deterministic answers grounded in public data sources. This ensures that completing the tasks requires constructing structured databases and performing precise analysis.

We propose DramaBot in Section 4, a multi-agent system built on top of Drama, comprising a data retriever and a data analyzer. The data retriever coordinates the execution of three sub-agents: web browser, data transformer, and web augmenter. The web browser collects raw data D through fine-grained webpage access. The data transformer then constructs a structured database T by extracting and organizing relevant information from D. If T does not include sufficient information, the web augmenter is triggered to collect D from a broader range of webpages in large quantities. The data analyzer defines and executes a function $f(Q, T) \rightarrow A$, where T is a structured table and A is the final answer to the user's query Q.

We evaluate DramaBot together with five baseline agents on DramaBench. DramaBot achieves 86.5% task accuracy at a cost of $0.05, outperforming all baselines with the highest accuracy and lowest API cost. The accuracy of DramaBot is 6.9× that of the recently released OpenAI Research Agent [49], while incurring less than 1/6 of the cost. DramaBot also demonstrates stable performance across different task categories and types. We report the details of these evaluations in Section 5.

The contributions of this paper can be summarized as follows:

- We propose an end-to-end paradigm, Drama, that unifies data collection, transformation, and analysis.
- We collect a benchmark, DramaBench, consisting of real-world tasks that are representative of Drama.
- We develop a multi-agent system, DramaBot, that effectively operationalizes Drama.
- We quantitatively evaluate the performance of DramaBot and state-of-the-arts agentic systems on DramaBench.

2 Drama: A Paradigm that Unifies Data Collection, Transformation, and Analysis

In this section, we introduce the Drama paradigm, a pipeline that takes as input a user query Q and returns an answer A by retrieving and analyzing data from open domains. We outline three major stages of Drama: data collection (Section 2.1), data transformation (Section 2.2), and data analysis (Section 2.3). For each stage, we illustrate the expected system performance, outline the necessary capabilities, and highlight the limitations of existing systems to demonstrate why Drama cannot be achieved through a trivial integration of existing systems and engineering effort. While we present Drama as executing a single iteration of these stages, it can be naturally extended to a multi-hop setting, including partially multi-hop workflows where, for example, the system iterates between data collection and data transformation before proceeding to analysis (e.g., $(1 \rightarrow 2)_n \rightarrow 3$ in Section 4), for more complex tasks.

We use two motivating examples of real-world tasks that require retrieving up-to-date data and demand accuracy: one example is answering the question: "Which (USA) state has the highest rate of homelessness in 2024?"[68] (Q_1). Another example arises from the proliferation of claims on social media, where individuals must quickly distinguish truth from misinformation, such as verifying: "Change in the number of births from 2022 to 2023 for Ireland: -10.3%"[77] (Q_2).

2.1 Data Collection

The `collect` function takes a user query Q in natural language as input and searches for relevant raw data D from open domains to answer Q. We present two examples in Figure 2, where to answer Q_1 and verify Q_2, the `collect` function browses open domains, identifies websites containing the most relevant and authoritative data, retrieves the raw data, and caches it as an intermediate result. Note that D can appear in various formats depending on the data source. In Figure 2, we see that for Q_1, the raw data D_1 is an Excel sheet [65], while for Q_2, D_2 is a report in PDF format [6].

Existing automated data science tools lack the capability to perform open-domain data search and retrieval. This functionality cannot be trivially achieved by combining information extraction or web search tools, as these systems are primarily designed for text-based lookups with limited context, rather than retrieving large-scale data from diverse sources and formats, as required by the `collect` function.

2.2 Data Transformation

The retrieved data D can appear in multiple formats, ranging from unstructured data like PDFs to structured data like CSVs. In this stage, the `transform` function extracts the required information from D and organize it as a structured database T. Specifically, DRAMA adopts a single-table representation in the data transformation step to enable more reliable execution of complex queries. Prior work has shown that directly querying over multi-table databases is already challenging and error-prone in closed-domain settings [34], and this difficulty is amplified in open-domain scenarios where data is less structured and consistent. Importantly, the process of integrating multiple tables often reveals the need for additional transformations that are not obvious when viewing each table in isolation, such as pivoting, transposing, unpivoting, or invoking user-defined functions. Such transformations are cumbersome and unnatural to express directly within downstream SQL query generation, and are more effectively handled in a dedicated transformation step prior to analysis.

As we show in Figure 2, D_1 is an Excel sheet containing 18 tabs, each with 1,306 columns. The `transform` function selects the relevant information, specifically the tab for the year 2024 and two columns (*State* and *Overall Homeless*), and produces T_1, a structured table containing these columns. Similarly, D_2 is a PDF report containing statistics on births, deaths, and natural increases from 2013 to 2023. The `transform` function extracts the birth data for 2022 and 2023 from a diagram and organizes it as T_2.

As an intermediate component between open-domain data collection and data analysis, the `transform` function must process data in diverse formats, parse them into structured tables, understand their relationships, and present them in forms that are interpretable and suitable for downstream analysis. Existing systems lack the capability to automatically handle such heterogeneous data formats in a unified and generalizable manner.

2.3 Data Analysis

This stage abstracts the task of answering a natural language query Q over a structured table T as the `analyze` function. In Figure 2, we demonstrate one approach: translating Q into SQL operations and executing the corresponding code. However, this stage is flexible and can be adapted to any other state-of-the-art semantic query parsing and execution systems, including translating Q into any other programming languages such as into Python code with pandas functions or using table question answering frameworks.

While the `analyze` function can be realized by a wide range of natural language query systems over structured tables, it also requires adaptation to handle the diverse and complex formats and content structures produced during earlier stages of the pipeline.

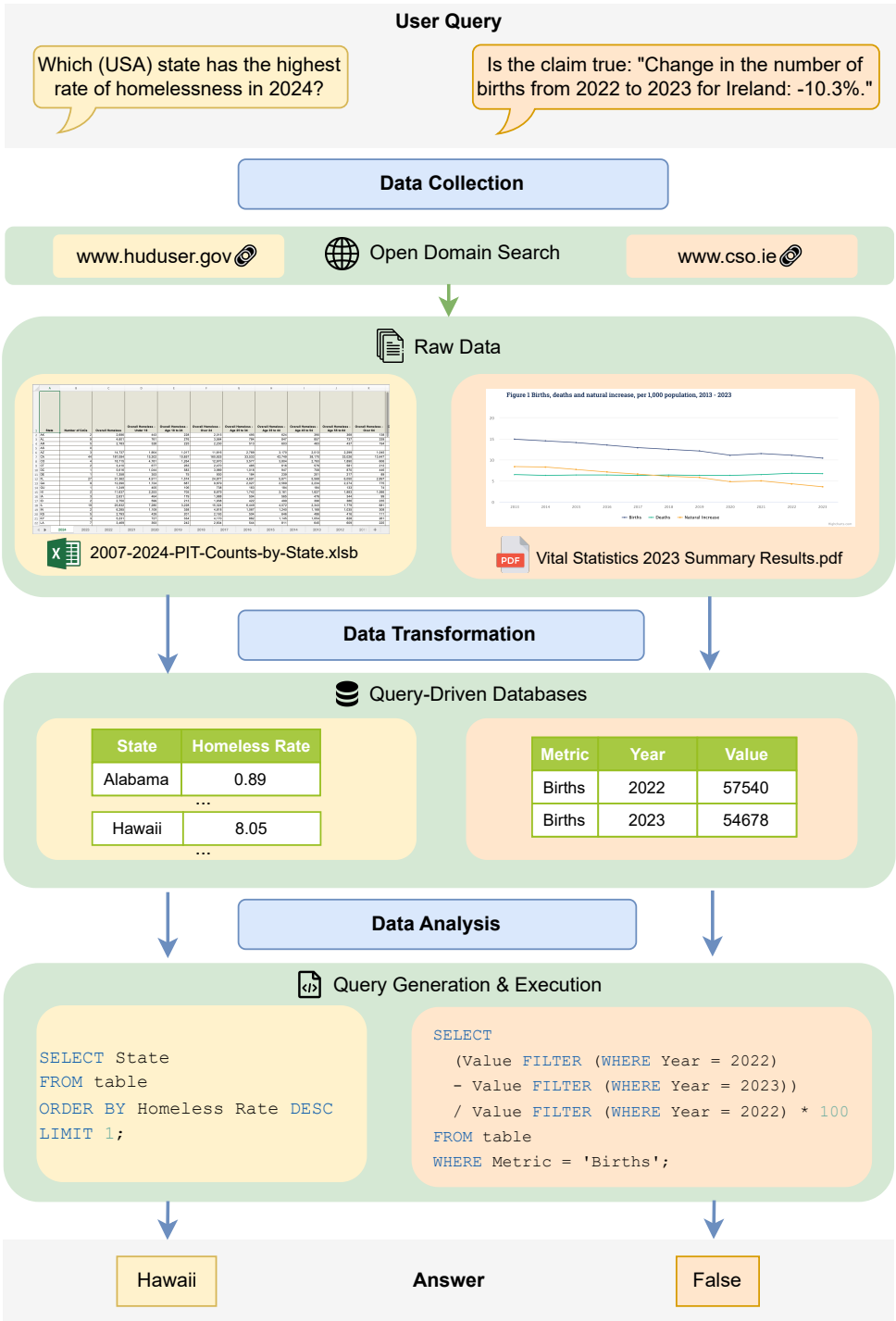


Fig. 2. Overview of the DRAMA paradigm. Here we present two examples: *(left)* user query as a question (Q_1), and *(right)* user query as a claim to be verified (Q_2).

3 DRAMABENCH: A Collection of Real-world DRAMA Applications

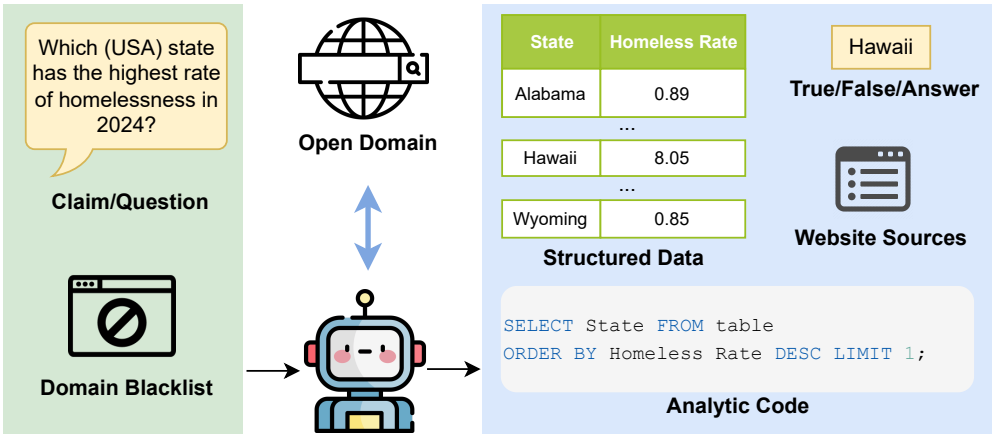


Fig. 3. Overview of each DRAMABENCH task. Given a user query, the agent is tasked with collecting, structuring, and analyzing data from open domains to generate an answer.

To quantitatively assess system performance on real-world tasks that faithfully reflect the core features of DRAMA, we construct DRAMABENCH, a benchmark comprising two task categories: (1) claim verification and (2) question answering, each with 100 task instances. We describe the task collection criteria and process in Section 3.1 and formally define the tasks in Section 3.2.

Table 1. Examples of DRAMABENCH data

Task	Claim/Question Example	Label
Claim Verification	Change in the number of births from 2022 to 2023 for Ireland: -10.3% [77]	False
	Change in the number of births from 2022 to 2023 for Ireland: -5% [76]	True
Question Answering	Which state has the highest rate of homelessness in 2024? [68]	Hawaii
	What is the average number of wildfires in the U.S. annually from 1983 to 2023? [67]	70,000

3.1 Task Collection

To ensure that DRAMABENCH reflects the core characteristics of DRAMA-style tasks and enables generalizable evaluation of agent performance in realistic, paradigm-aligned scenarios, we apply the following universal criteria to both task categories: (1) the claim/question should be real-world and important (i.e., publicly posted on influential social media platforms and significant enough to warrant verification or answering); (2) it should be analytic in nature (i.e., based on data analysis results); (3) it should be non-vague (i.e., clearly and deterministically mappable to the execution result of a SQL script); (4) it should be based on publicly available and verified data, with explicit reference to the data.

We further introduce task-wise data collection criteria as follows:

Claim Verification. In this task, the agent is required to retrieve data from open domains to verify whether a claim is true or false. To ensure a fair and balanced comparison, we require that true and false claims exhibit: (1) a comparable number of instances, and (2) comparable difficulty

levels, both in data retrieval and data analytics. To do this, we source claims from PolitiFact [71] and Twitter/X Community Notes [9]. As a task-specific criterion, we require that the falsity in claims must originate from incorrect data analysis.

For PolitiFact, we extract all 18 fact-check posts with ratings equal to or lower than Mostly False, published between 01/01/2024 and 02/28/2025, that satisfy the universal and task-specific criteria. From each post, we collect the original claim (as the false claim) and the accompanying explanation (as the true claim).

For Twitter/X Community Notes, we download all notes available as of 02/28/2025 and apply the following filters: (1) the original Twitter/X post is classified as `MISINFORMED_OR_POTENTIALLY_MISLEADING`; (2) the post contains a `misleadingFactualError`; (3) the note refers to `trustworthy Sources`; (4) the `currentStatus` of the community note is `CURRENTLY_RATED_HELPFUL`; (5) the post contains no media contents (e.g., images or videos); (6) the note includes the keyword `data`; (7) both the post and the note are in English; and (8) to ensure data stability due to the fast pace and loosened constraints of community notes, we restrict collection to notes posted between 01/01/2024 and 01/01/2025, ensuring a minimum 3-month gap from the time of collection. This filtering yields 545 pre-filtered notes. From these, we manually extract 32 that satisfy all universal and task-specific criteria. We treat the Twitter/X post as the false claim and the community note as the true claim.

Our pairing strategy in collecting claim verification tasks ensures a balanced number and comparable difficulty, as both claims are based on the same dataset and share the same underlying SQL logic. We provide example true and false claims in Table 1.

Question Answering. In this task, the agent must retrieve data from open domains to answer a given question. To ensure reliable evaluation, we impose the following task-specific criterion: the answer must be representable as a single string or numeric value resulting from data analysis, such as the output of a SQL query or an equivalent computation.

We collect all 100 valid statements from the Economy and Education sections of USAFacts [70], published between 01/01/2024 and 02/28/2025, that satisfy the universal and task-specific criteria. We use GPT-4o [46] to rephrase each statement into a question and then manually annotate the ground-truth answers based on the original USAFacts article. Table 1 includes examples with both text and numeric answers. There are 31 task instances with text answers and 69 with numeric answers as ground truths.

For each task instance for both verification and QA tasks, we collect the ground-truth answer, manually transform the original source data into a structured table, and annotate SQL scripts that reproduce the result. This enables fine-grained evaluation of both data retrieval and analysis performance. Specifically, all 200 task instances were annotated by a team of 5 researchers, each with substantial SQL experience. The annotation process consisted of two stages:

(1) Each annotator independently transformed the raw data into structured form and generated the corresponding SQL code in response to the natural language question or claim. At this stage, each data-code pair was executed to ensure it produced the expected ground-truth answer (i.e., True/False for verification tasks or the exact value for QA).
(2) The remaining four authors then independently cross-verified the data transformation and SQL code. Any discrepancies were resolved collaboratively until full agreement was reached on the final version.

To prevent data leakage, we compile the original sources into a blacklist, which is provided to the agents at execution time. Agents are explicitly restricted from accessing any content in the blacklist during task execution. We verify that, although not explicitly constrained, for all 200 out

of 200 tasks, all data required to answer each query, while potentially spanning multiple files or tables, comes from a single domain by the nature of the tasks.

3.2 Task Definitions

As we illustrate in Figure 3, we formally define the DRAMABENCH tasks as follows: given (1) a claim to be verified or a question to be answered, and (2) a blacklist of domains that must not be used as data sources, the system is required to retrieve relevant data from open domains (e.g., the web) and analyze it to output: (1) the validity of the claim or the answer to the question, (2) the retrieved data, transformed into a structured table, (3) the analytic code used to derive the result, and (4) traces (i.e., the websites accessed during the data collection process).

4 DRAMABOT: Realizing DRAMA Through a Multi-Agent System

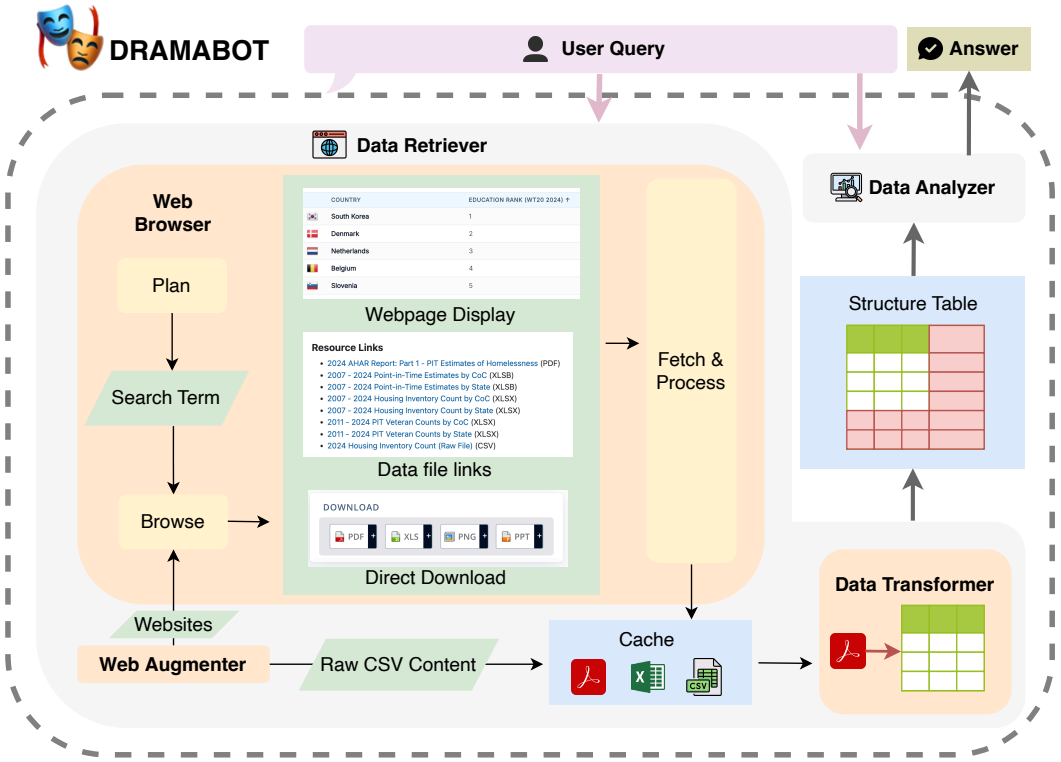


Fig. 4. Overview of DRAMABOT.

In this section, we develop DRAMABOT following DRAMA. DRAMABOT is a multi-agent system that is capable of executing the end-to-end data science pipeline consisting of a data retriever (Section 4.1) that executes stages 1 and 2, data collection and data transformation, and a data analyzer (Section 4.2) that executes stage 3, data analysis.

Following the DRAMABENCH task interface, DRAMABOT takes as input a user query Q and a blacklist of domains B that should not be accessed, and returns: the answer A to the query, the structured table T containing information to answer the query, the set of sources S used to obtain the data, and the code program P executed on T to generate A. DRAMABOT ensures that no source $s \in S$ is drawn from $\mathcal{B}$.

4.1 Data Retriever

The goal of the data retriever is to: **(G1)** collect data D sufficient to answer Q, **(G2)** ensure the data comes from reliable sources S that are not associated with B, and **(G3)** transform D as a structured table T tailored to Q, stored in DramaBot environment shared with data analyzer.

The data retriever coordinates the execution of three sub-agents: web browser (Section 4.1.1), data transformer (Section 4.1.2), and web augmenter (Section 4.1.3). We provide an overview of the workflow in Figure 4. The data collection stage begins with an empty set of sources $S = \emptyset$, and progressively builds S as the data retriever coordinates the web browser and the web augmenter as two complementary modules. The web browser performs detailed, step-by-step exploration to collect data from a small and precise set of sources, aiming to identify the most reliable domain. If it fails to collect sufficient information from authoritative sites, the web augmenter retrieves a broader set of candidate data from less curated sources in large quantity. This data is then inspected based on both source reliability and content relevance to ensure accuracy. Through this coordination, the data retriever ultimately selects a single, most trustworthy domain and retrieves all required data from that source. The data transformation stage is performed by the data transformer, where the data collected by the web browser and the web augmenter is aggregated and organized. The key distinction of DramaBot from existing systems lies in its ability to collect large-scale data across a wide range of formats using the web browser, enabled by its uniquely designed interfaces with open domains and coordination with the web augmenter. Additionally, the data transformer plays a critical role in converting this heterogeneous data into a single, descriptive structured table suitable for downstream analysis.

We now describe the structure, communication, and invocation conditions of each agent in detail.

4.1.1 Web Browser. The web browser takes Q and B as input and collects raw data D, stored in cache as intermediate results. A starting website, W, can also be provided as input; by default, $W = \emptyset$, meaning the web browser starts on the `google.com` page and generates an initial search term to guide the search path for data retrieval.

The browsing process is adapted from WebVoyager [23], a general-purpose web agent. The web browser uses Selenium [55], and at each step, the agent receives a screenshot of the webpage as an observation of the environment, along with feedback from previous steps. For every iteration, the web browser selects an action from its action space and executes it in the web environment. Before selecting an action in each round, the URL of the driver task for that round is added to S.

We further make the following modifications to its original action space, which includes `Click`, `Type`, `Scroll`, `Wait`, `GoBack`, `Google`, and `ANSWER`:

(1) Modify the `Click` action: this action only clicks elements that are not associated with any domain in B.
(2) Remove the `Answer` action.
(3) Add `GetData` action: this action completes the browsing process by returning the specific data contents in CSV format and should be performed when the agent finds the necessary data to answer Q.
(4) Add `CheckLink` action: this action checks the hyperlink of a web element.
(5) Add `GetLink` action: this action completes the browsing process by returning the hyperlink to the data file.
(6) Add `Download` action: this action monitors the data file download process and completes the browsing process if the data file is successfully downloaded.

Adjustment (1) supports **(G2)** by preventing the web browser from accessing unreliable sources. Adjustments (2)–(6) support DramaBot's ability to meet **(G1)** by enabling large-volume data

exchange between open domains and the local environment, overcoming the limitations of existing web search tools. Specifically, we replace the simple textual response (Adjustment (2)) with the three data collection mechanisms (Adjustments (4)–(6)) to accommodate the following variety in data sources:

First, data can be directly collected from webpages (Adjustment (3)). For example, as we show in Figure 4, to verify the claim "The USA ranks #1 in education rankings in Education Rankings by Country 2025,"[78] the relevant rank data is directly displayed on the webpage [75]. Second, data can be collected via direct hyperlinks exposed in the webpage HTML (Adjustments (4) and (5), which operate as a pair). For example, as we show in Figure 4, to answer Q_1, the hyperlinks to the relevant data files are visible and embedded directly in the page [65]. Third, data can be collected from files that are served through backend JavaScript pipelines, which makes them difficult to extract directly from the HTML content. For example, as we show in Figure 4, verifying the claim "India average cigarettes inhaled daily: 8"[79] requires clicking a download button [57] and waiting for the file to finish downloading before the relevant data becomes accessible.

Corresponding to the action space, we provide a list of updated action guidelines to ensure accurate and consistent behavior of the web browser. Below, we present four critical instructions:

(1) `GetData` should browse the webpage incrementally, view by view, and gradually accumulate raw data into a structured table with clear and straightforward column names until sufficient information is collected. This strategy is supported by the promising performance of existing multimodal large language models (MLLMs) on data extraction tasks [47], which enables reliable extraction from each individual view (i.e., webpage screenshots).
(2) `CheckLink` must be executed before performing `GetLink` on the same element. This ensures that all returned hyperlinks are verified as pointing to valid data files.
(3) The web browser should always scroll up and down to view the entire webpage before deciding to navigate to other pages. This ensures that the web browser thoroughly explores each page before proceeding.
(4) The web browser should prioritize retrieving data from authoritative sources, such as official government websites, and avoid low-quality or unofficial domains. This helps prevent misinformation from unverified sources.

The browsing process returns collection targets R for raw data D, which appear as (1) raw contents (via `GetData`), (3) hyperlinks to data files (via `GetLink`), and (3) original data files (via `Download`). The web browser further processes R to store D as cache in agent environment. Specifically, the web browser issues the following shell commands via conventional software pipelines:

(1) **Raw content:** Dump the raw contents R as a CSV file in the agent environment:

```
echo $R > /agent_env/cached_data.csv
```

(2) **Hyperlink:** Fetch the file from the hyperlink R using `curl`, and unzip if necessary:

```
curl -OJ $R --output-dir /agent_env/
unzip /agent_env/*.zip
```

(3) **Original file:** Move the downloaded file R from the download directory into the agent environment:

```
mv /downloads/$R /agent_env/
```

With D collected, the web browser concludes its task round and returns control to the data retriever, which then invokes the data transformer (Section 4.1.2).

4.1.2 Data Transformer. The data transformer performs stage 2 of Drama, where the raw data D is transformed into a structured table T based on Q for subsequent analysis to support **(G3)**.

Before introducing the detailed operations carried out by the data transformer, we define its core function: `aggregate_tables(`T_1`, `T_2`, `Q`, . . . )` $\rightarrow T_{\text{new}}$. Unlike traditional SQL join methods that follow fixed syntax and join logic, this function supports a variety of merge strategies that adapt dynamically to Q and other given contexts. We summarize three basic patterns of table aggregations used when constructing T. We use C to denote columns and R to denote rows. Specifically,

- **Column Aggregation**: Combines attributes from multiple tables along shared rows based on a common key. For example, T_1 contains homelessness rates by state, T_2 contains housing inventory counts by state, and Q asks for the correlation across the factors in the two tables [68]. In this case, joining T_1 and T_2 is equivalent to performing a SQL join on the shared `State` column:

```
SELECT * FROM T1
JOIN T2 ON T1.State = T2.State
```

- **Row Aggregation**: Merges rows from two tables with the same schema to form a single dataset spanning a larger range. For example, T_1 contains the birth rates in Ireland from 2003–2013, T_2 contains the same data from 2013–2023, and Q asks for trends across these two decades [77]. In this case, joining T_1 and T_2 involves stacking T_2's rows below T_1's, corresponding exactly to `UNION` or `UNION ALL` in SQL:

```
SELECT Year, BirthRate FROM T1
UNION ALL
SELECT Year, BirthRate FROM T2
```

- **Mixed Aggregation**: Combines row-wise merging with the addition of distinguishing metadata to track source-specific attributes. For example, T_1 contains homelessness rates by state for 2023, T_2 contains the same for 2024, and Q asks to compare the rates across the two years [68]. To support this, a new column `Year` is added, with 2023 assigned to rows from T_1 and 2024 to rows from T_2. All rows are concatenated and preserved in T_{new}:

```
SELECT state, rate, 2023 AS year FROM T1
UNION
SELECT state, rate, 2024 AS year FROM T2
```

In addition to following these basic patterns, the `aggregate_tables` function is also instructed to apply transformations that facilitate downstream analysis, such as renaming columns with contextually meaningful names, based on the specified context, including Q.

We now illustrate the workflow of the data transformer along with the example from Section 1, with the National Park Visitor Spending Effects report downloaded locally as a PDF, as part of D. The data transformer maintains a list L to track the files it has already inspected. To assist in decision-making, the contents of any file whose name contains `readme` (case-insensitive) are first parsed and incorporated into the context. Once parsed, these readme files are added to the checked file list L. Starting with $T = \emptyset$, the transformer performs the following operations in each iteration:

`check_adequate_info(`Q`, `T`)` $\rightarrow$ `{(True, `C`), (False, `M`)}`. This operation checks the adequacy of existing data through a code-driven inspection process. The data transformer loads the current local data storage T and determines whether it contains sufficient information to answer the query Q. To do this, it attempts to generate an executable code snippet based on T to answer Q. If a deterministic code snippet can be generated, this indicates that T is adequate. The data transformer then completes the current task round and returns C as a reference for coordination

with other agents in DRAMABOT. If not, it outputs a list of missing information M to guide future decisions on file selection and information extraction, i.e., the proceeding operations. In the first iteration of our example, this operation returns `False` because T is empty, identifying the park names and total visitor spending as the missing information M.

`file_selection`$(Q, M, D, L) \rightarrow F$. This operation selects a data file F from the set of unprocessed files $D \setminus L$ that is most likely to contain the missing information M. The data transformer is presented with a list of file names. In our example, the National Park Visitor Spending Effects report file is selected because its name is the most relevant and likely to contain the missing information.

`extract_data`$(Q, M, F, L) \rightarrow T'$. This operation extracts the information necessary to answer Q, particularly the missing elements in M, from a data file F, and formulates it as a tentative structured table T'. We categorize the following two extraction approaches based on the data type of F:

- **Structured data** (e.g., `.csv`, `.tsv`) and **Semi-structured data** (e.g., `.xls`, `.xlsx`): The structured units of F (e.g., the entire CSV file, or each sheet of an Excel file) are first extracted as $T_1, T_2, \ldots, T_n$. These tables are then iteratively joined using `aggregate_tables` with both Q and M passed in contexts: at each step i, the intermediate result T'_i is updated as $T'_i = \texttt{aggregate\_tables}(T'_{i-1}, T_i, Q, M)$, starting with $T'_0 = \emptyset$ for T_1, including the sole structured data file, to apply transformations.
- **Unstructured data** (e.g., `.pdf`): Given the robust performance of MLLMs in structured data extraction from various document types [3], we use MLLMs (GPT-4o [46]) for data extraction. However, since MLLM performance tends to degrade with increased input volume [39], our core strategy is to extract information from small, manageable portions of data incrementally and then aggregate the results. Specifically, the minimal processing unit of F (e.g., a page from a PDF) is sequentially passed to the MLLM. Starting with an empty dataframe $T' = \emptyset$, the MLLM is instructed to iteratively expand T' with the necessary information to answer Q and addresses the information shortage included in M, stopping once all required information is captured or the entire file has been processed. To overcome the context limitations of MLLMs, T' is included in each API call as context. To overcome the form-generation limitations of MLLMs, the model is instructed to return T' containing only the minimal information necessary and relevant to answering Q.

At the end of this operation, before returning T', F is added to L. In our example, since the report is a PDF file, the MLLM is invoked on each page. As the first 26 pages do not contain relevant information, T' remains empty. Starting from page 27, where the table first appears, T' is incrementally enriched with the table contents (i.e., the rows containing park and visitor spending information), continuing until the table concludes at page 39, the operation would return and add the report to L.

`update_database`$(Q, M, T, T') \rightarrow T$. In this operation, the extracted tentative structured table T' is joined with T to enrich the available information. Specifically, T is updated as $T = \texttt{aggregate\_tables}(T, T', Q, M)$. In our example, T' extracted from the report is aggregated with an empty table, where its column names are reformatted to include necessary dashes or underscores to facilitate downstream analysis.

Once all files have been checked in all iterations, the data transformer concludes its task round and returns control to the data retriever. The data retriever then verifies whether T is nonempty and valid, i.e., whether it contains sufficient information to answer Q. If this condition is met, the data retriever concludes its task and hands control over to the data analyzer (Section 4.2). If not,

the data retriever invokes the web augmenter (Section 4.1.3), triggering additional iterations of stages 1 and 2 of the DRAMA paradigm. In our example, when `check_adequate_info` is called in the second iteration, it returns `True` together with the SQL code snippet we show in Section 1, concluding the process.

4.1.3 Web Augmenter. The main goal of the web augmenter is to complement the fine-grained searches of the web browser, where large crowds of sources are fetched within a short timeframe using parallel workloads to support **(G1)**. Specifically, the web augmenter uses the OpenAI search tool [51], and given Q with access restrictions to B, it is tasked with searching the websites to produce data D' containing specific data contents in CSV format using clear and straightforward column names that are necessary to answer Q. The sources S' accessed by the web augmenter are included in the `annotations` field of the response.

Although explicitly prompted not to access domains in B, when using an integrated tool that doesn't strictly follow the prompts, the web augmenter may still access them. Thus, to support **(G2)**, when the web augmenter concludes its task round and returns control to the data retriever along with its collected outputs, the data retriever must decide whether to trust the data collected from these large-crowd sources based on their S', or initiate another round of web browsing using the reliable sources $S'' = S' \setminus B$. The data retriever processes the outputs of the web augmenter as follows:

If $s \notin B, \forall s \in S'$, the sources S are augmented with $S = S \cup S'$ and the data in cache is augmented with $D = D \cup D'$. The data transformer is then called to process D', and once it returns T, the data retriever concludes its task.

If $\exists s \in S'$ s.t. $s \in B$, the data retriever performs the following steps:

(1) Excludes all the sources associated with the blacklisted domains by constructing $S'' = S' \setminus B$.
(2) The data retriever calls the `rank_website` function to rank S'' from the most relevant and reliable sources to answer Q to the least, based on Q and the response of the web augmenter, including the result, data, and code, with inline annotations where each $s \in S''$ is used to generate the response. `rank_website` ranks the reliability and relevance of websites according to their contributions to the response to the query generated by the Web Augmenter.
(3) The web browser is called iteratively with the sources $s \in S''$ passed as the starting website W, following the order of the website rankings. For each iteration i:
    - D is expanded with the newly collected data D_i as $D = D \cup D_i$, and S is expanded with the new sources accessed as $S = S \cup S_i$.
    - The data transformer is called to process D_i. Once the data transformer outputs valid storage T, the data retriever concludes its task, and DRAMABOT calls the data analyzer (Section 4.2).

4.2 Data Analyzer

With a structured table T stored in local storage, the data analyzer further answers Q over T. When $T = \emptyset$, the analyzer falls back to return `False` for verification tasks, indicating that no data are available to support the claim and defaulting to an educated guess, which in many real-world verification scenarios corresponds to labeling unsupported claims as false. When $T \neq \emptyset$, we implement an NL2SQL system by prompting `gpt-4o` [46] as the query generator. Building on the strong performance of existing LLMs on coding tasks [8], our key innovation lies in making the model aware of diverse and potentially noisy data retrieved from open-domain sources. The query generator takes the table head as input to observe and adapt to unconventional or noisy data contents through few-shot learning [4]. The generated code is executed over T to produce the answer A generated by DRAMABOT in response to Q.

5 Experiments

We introduce the experimental setup in Section 5.1. We report the overall system-level performance in Section 5.2, and examine their generalizability in Section 5.3. We analyze the stage-wise performance of all the agents in Section 5.4 and evaluate the impact of DRAMABOT's subagents, including ablation studies, in Section 5.5.

5.1 Experiment Setup

We evaluate the performance of DRAMABOT and five baseline agents on DRAMABENCH.

5.1.1 Baselines. We select the following five baseline agents that have demonstrated strong performance across a variety of real-world tasks:

Deep Research [61]. Deep Research is an agent that integrates planning, searching, reasoning, and reporting, and has shown strong performance on research tasks involving the collection and analysis of complex document corpora. We select Deep Research to examine how its capabilities generalize to tasks requiring the retrieval and analysis of complex, large-scale data. We use the open-sourced version of the Gemini Deep Research reasoning framework [19].

AutoGPT [21]. AutoGPT is a multi-functional agent designed to handle a wide range of tasks, featuring capabilities such as web search, long-term planning, tool selection and usage, and self-reflection. We include AutoGPT to evaluate how a general-purpose agent performs on DRAMABENCH in comparison to DRAMABOT.

WebVoyager [23]. WebVoyager is an MLLM-powered web agent capable of executing user instructions end-to-end by interacting directly with real-world websites. We select WebVoyager to evaluate the effectiveness of agents specifically designed for web-based tasks in retrieving and analyzing large-scale data from open domains. Since the web browser component of DRAMABOT builds upon that of WebVoyager, it also serves as a baseline for comparing the design of the action space and reasoning mechanisms of DRAMABOT.

OpenAI Research Agent [49]. The OpenAI Research Agent is a recently released system introduced alongside OpenAI's new agent framework, which received more than 8.6K stars on GitHub within a month. It consists of a planner agent for generating search terms, a search agent for retrieving relevant information, and a report writing agent for composing answers. Since its search agent employs the same OpenAI web search tool used by the web augmenter in DRAMABOT, it provides a natural point of comparison for in terms of the architectural design and coordination strategies of DRAMABOT.

OpenAI Web Search Tool [51] + TAG [2] (Search + TAG). To explicitly demonstrate that DRAMA extends beyond trivial integration of existing systems, we implement a baseline system composed of the OpenAI WebSearch Tool, representing the state of the art in **(C1)**, combined with the best-performing version of TAG using LOTUS, which supports both **(C2)** and **(C3)**. The search tool is executed for 10 iterations, with each iteration refining and extending the previous response, matching DRAMABOT 's maximum number of iterations for a fair comparison.

All agents, including DRAMABOT and the five baselines, are primarily powered by `gpt-4o-11-06`, with one explicit original design exceptions: the OpenAI Research Agent's writing agent uses `o3-mini` [48]. All agents are explicitly instructed not to access domains on the blacklist. For agents that support software-enforced blocking (e.g., via DNS), we apply such restrictions directly. Specifically, WebVoyager is prohibited from clicking links from blacklisted domains, and Google Custom Search API used by Deep Research is configured to exclude results from these domains.

Unless otherwise specified, agents follow the I/O requirements of DramaBench. The only exception is Search + TAG, which does not generate explicit code and therefore produces only outputs (1), (2), and (4) as defined in Section 3.2.

5.1.2 Metrics. We apply two categories of metrics: system level and stage level, to quantitatively evaluate agent performance and cost efficiency on DramaBench.

System-level Metrics. We deploy accuracy as our primary performance metric, which measures whether the agent's output (i.e., output (1) per definition in Section 3.2) matches the ground-truth label. Any use of blacklisted domains is considered an automatic failure. To assess whether the agents' outputs are grounded in data retrieval and analysis, we evaluate the coherence among the retrieved data, the generated code, and the final result: specifically, whether the execution result of the generated code on the retrieved data aligns with the ground-truth label, namely the *data-grounded (DG) accuracy*. For cost analysis, we report the average API costs for all requests made by each agent for each task.

Stage-level Metrics. We evaluate the performance of different Drama stages by assessing the quality of their intermediate results. Specifically, we examine the quality of the retrieved data (output of stages 1-2) and the generated code (output of stage 3). For data, we assess (1) data validity, i.e., whether the agent successfully collects data from open domains and organizes it into CSV format as a structured table, and (2) data similarity, specifically, schema similarity, with the ground-truth data. For code, we evaluate (1) the execution success rate, i.e., whether the code compiles, executes without error, and generates a result, and (2) code similarity with the ground-truth code. Since there is no universally accepted gold standard for evaluating data and code similarity, we apply various evaluation methods to automate this process in a quantitative, comprehensive, and objective manner. We evaluate the performance of different Drama stages by assessing the quality of their intermediate results. These similarity metrics are included primarily as diagnostic checks to better understand system behavior.

Table 2. Different similarity metrics applied to data and code.

	LLM as a Judge	**Embedding**
w/o Column Match	$DataSim_1$	$DataSim_2$
w/ Column Match	$DataSim_3$	$DataSim_4$
w/o Normalization	$CodeSim_1$	$CodeSim_2$
w/ Normalization	$CodeSim_3$	$CodeSim_4$

We summarize the similarity metrics in Table 2. We use two backbone methods to evaluate similarity: (1) LLM-as-a-judge: For data, we input `df.head()` for both the agent-retrieved and ground-truth data into `gpt-4o-mini` [45], prompting it to generate a similarity score between 0 and 1. For code, we follow the same procedure: both the agent-generated and ground-truth code are provided to the model to generate a similarity score. (2) Embedding-based similarity: For data, we encode `df.head()` using `text-embedding-3-small` [50], an embedding model that effectively captures tabular structure and content in text format. For code, we use CodeBERT [16], a model trained on large-scale code corpora and effective at capturing functional and structural similarities in code. We then compute the cosine similarity between the generated embeddings to obtain a numerical similarity score.

Given the nature of structured data and code, we apply alignment techniques to account for structural and semantic variability in similarity evaluation. For data, since it consists of columnar structures, we perform *column-level matching*. Each column is individually encoded using `text-embedding-3-small` [50], and then matched to the most similar column in the corresponding ground-truth data based on cosine similarity of the embeddings. For code, we implement a 3-stage custom normalization pipeline that standardizes both variable naming and symbolic expressions before measuring similarity: (1) Parse the code into Abstract Syntax Trees (ASTs) [54] (2) Rename user-defined variables to canonical forms (e.g., `var_0`, `var_1`, etc.), while preserving built-in names (3) Uses sympy to expand and simplify mathematical expressions in binary operations (e.g., both `x = (a + b) * (a + b)` and `x = (a + b) ** 2` are expanded to `a**2 + 2*a*b + b**2`) to preserve only the logical structure and semantics. This normalization is deterministic and converges to a unique normal form for a given input under the defined transformation rules.

5.2 DramaBot Achieves Promising Performance At Low Costs

Table 3. System-level performance and costs of different agents. Accuracy (Acc.) and Data-Grounded Accuracy (DG Acc.) are reported in percentages (%), and cost is measured in $0.01 units. The optimal value for each metric is highlighted with both bold and underline formatting.

Agent	Overall			Verification			QA		
	Acc.↑	DG Acc.↑	Cost↓	Acc.↑	DG Acc.↑	Cost↓	Acc.↑	DG Acc.↑	Cost↓
Deep Research	32.5	27.5	8.24	37	29	8.97	28	26	7.51
AutoGPT	21.5	4.0	9.36	29	6	9.53	14	2	9.19
WebVoyager	46.5	20.5	5.29	49	16	**6.33**	44	25	4.26
OpenAI Research Agent	12.5	12.5	33.94	12	12	33.95	13	13	33.93
Search + TAG	25	-	9.31	40	-	8.67	10	-	9.94
DramaBot	**86.5**	**82.5**	**5.03**	**88**	**80**	7.62	**85**	**85**	**2.45**

Table 4. Accuracy (%) of baseline agents without enforcing data and code generation under the DramaBench I/O format, compared with DramaBot's performance. The optimal value for each metric is highlighted with both bold and underline formatting.

Task	Deep Research	AutoGPT	WebVoyager	OpenAI Research Agent	DramaBot
Overall	19	17.5	28.5	7.5	**86.5**
Verification	33	29	52	12	**88**
QA	5	6	5	3	**85**

As we can see from Table 3, DramaBot achieves 86.5% accuracy on all DramaBench tasks with an average cost of $0.05, outperforming all the baseline agents across all metrics. We demonstrate our findings in detail as follows.

DramaBot significantly outperforms existing AI agents on DramaBench. We observe that existing AI agents show limited capabilities in performing DramaBench tasks. Specifically, all baseline agents fail on more than half of the DramaBench tasks, with accuracies ranging from as low as 12.5% to a maximum of only 46.6%, making them unreliable for practical use. Notably, in verification tasks where the expected accuracy is 50% through random guessing, none of the baseline agents reach this level.

DramaBot achieves over 85% accuracy for both tasks, which is effective for practical deployment in real-world scenarios. It outperforms the baseline agents with an end-to-end accuracy of 2.7× as compared to Deep Research, 4× as compared to AutoGPT, a relative increase of 86% for WebVoyager, 6.9× as compared to the OpenAI research agent, and 3.5× as compared to Search + TAG.

DramaBot reliably grounds answers in data. When it comes to DG accuracy, the performance of baseline agents further deteriorates, with the highest DG accuracy reaching only 27.5%. For instance, while WebVoyager achieves a relatively high end-to-end accuracy of 46.5%, its DG accuracy drops to 20.5%, indicating that fewer than half of its correct answers are actually grounded in the retrieval and analysis of large-scale data. AutoGPT performs even worse, with only 4% of its outputs being both correct and grounded, which is less than one-fifth of its original accuracy. These results highlight that existing agents often fail to generate reliable, data-grounded answers. Instead, they may rely on superficial information found on the web, treating it as truth without assessing source reliability or rigorous data analysis.

In contrast, DramaBot achieves a DG accuracy of 82.5%, indicating high reliability. Its architecture ensures that final results are derived from the execution of analytic code over retrieved data. The 4% gap between its overall accuracy and DG accuracy reflects the fallback mechanism used when no relevant data is retrieved.

For the four baseline agents whose original outputs do not require generating data and code, we conducted ablated experiments following their native setups, where only the final result is required (i.e., they are not required to output data and code following the DramaBench I/O format). As we show in Table 4, their performance remains largely unchanged on verification tasks but drops significantly on question answering tasks, as the lack of data grounding leads them to make unreliable guesses. This further underscores the importance of the data grounding principles introduced by Drama.

DramaBot is cost-efficient in design. DramaBot incurs the lowest cost among all baseline agents, demonstrating its lightweight and efficient design.

First, in terms of architecture, traditional general-purpose agents like AutoGPT integrate complex modules with extensive planning and coordination logic, leading to significant overhead. In contrast, DramaBot adopts a streamlined architecture that retains only essential components required for data retrieval and analysis, resulting in an average cost that is nearly half that of AutoGPT.

Second, at the workflow level, existing agents tend to over-plan and execute redundant actions. For example, the OpenAI Research Agent generates an average of 8 search terms per task, resulting in a substantial cost overhead of 6.8×. Similarly, Deep Research issues an average of 3 queries per task, incurring a cost of $0.08, which is still notably higher than DramaBot 's more efficient $0.05. In contrast, DramaBot adopts a streamlined approach: it issues a single, focused query and follows through deeply. Although DramaBot is capable of iterating through stages 1->2 in a multi-round fashion, it rarely needs to do so in practice: on average, only 1.41 rounds are required to retrieve the necessary information, highlighting the precision of its initial query formulation.

Third, DramaBot 's reasoning mechanism is optimized for cost efficiency. Even when compared to WebVoyager, which operates within a similar web-based action space, DramaBot is more effective at extracting critical information. On average, DramaBot performs 4.4 iterations of web browsing, while WebVoyager performs 2.29 iterations. Despite deeper explorations that lead to better performance, DramaBot incurs lower costs due to its reasoning prompts, which are designed to guide the agent toward only the most essential information while minimizing unnecessary input and output tokens.

DRAMABOT addresses core deficiencies of existing baseline agents. By examining detailed traces of baseline systems, we identify their key limitations and demonstrate how DRAMABOT effectively addresses them:

Web search-based agents face challenges due to limitations in output diversity and length. Originally designed for simple point-lookups, these systems struggle with tasks requiring structured data manipulation. For example, WebVoyager can browse a webpage [58] listing state minimum wages but fails to answer analytic questions such as "How many states have minimum wages higher than the federal minimum wage in 2024?" [69], which require operations like `COUNT` over structured data. Similarly, when tasked with identifying "Which state has the highest rate of homelessness in 2024?" [68], both Search+TAG and AutoGPT return incomplete results, unable to process and analyze full documents. DRAMABOT overcomes these limitations with an integrated interface that downloads, transforms, and analyzes data of diverse formats in a modular manner.

Research agents generate search queries, retrieve articles, and summarize them. However, they often rely on high-level textual summaries rather than verifying claims using original data sources, introducing an additional failure mode: hallucination from unreliable secondary sources. For instance, when evaluating the claim "Uttar Pradesh is now the second-largest economy in the country, competing to become India's top economy" [27], Deep Research references secondary articles that propagate misinformation. In contrast, DRAMABOT grounds its reasoning in authoritative data [59], reflecting the robustness of DRAMA.

DRAMABOT produces interpretable and grounded end-to-end traces. Here we demonstrate how DRAMABOT follows a coherent, multi-stage reasoning trace to verify a real-world factual claim. In verifying the claim, "The Federation for American Immigration Reform estimated that there are 16.8 million illegal aliens in the United States in 2023" [64], DRAMABOT first generates the search query *"Federation for American Immigration Reform 16.8 million illegal aliens 2023"* and uses it to search on Google. It navigates to the official website of the Federation for American Immigration Reform [15], locates the relevant report for 2023, and downloads the associated PDF file [14].

During the data transformation stage, DRAMABOT identifies a plot on Page 1 of the PDF and transforms it into a structured CSV file with the schema: `[Year, Estimated U.S. Illegal Alien Population (millions)]`.

For the analysis stage, DRAMABOT generates a code snippet that retrieves the estimated population for the year 2023, semantically similar to the following SQL query:

```
SELECT "Estimated U.S. Illegal Alien Population (millions)"
FROM table WHERE Year = 2023
```

5.3 DRAMABOT Exhibits Reliable Stability

We analyze the performance of DRAMABOT and baseline agents across tasks with diverse features to assess their generalizability and stability.

DRAMABOT achieves stable performance across different task categories. Compared to verification tasks that require binary outputs, QA tasks demand more concrete answers, increasing the complexity of analytic reasoning involved. This added complexity is reflected in the accuracy drop observed in both DRAMABOT and the baseline agents. As we show in Table 3, nonetheless, DRAMABOT maintains strong and stable performance, achieving 85% accuracy with only three fewer successful executions than in verification tasks. Notably, DRAMABOT's data-grounded accuracy increases in QA tasks by 5 additional correct instances compared to verification, demonstrating its generalizability. In contrast, AutoGPT's performance drops by half when transitioning from verification to QA tasks, indicating reduced adaptability to more complex tasks.

Table 5. The accuracy of different agents across tasks with different ground-truth labels and formats. T represents True, F represents False, N represents numeric values, and S represents strings. δ denotes the relative difference between two subtypes within the same task, and is calculated as $\delta = \frac{|A-B|}{A+B} \times 100$. The optimal value for each metric is highlighted with both bold and underline formatting.

Agent	Verification			QA		
	T (%)↑	F (%)↑	δ (%)↓	N (%)↑	S (%)↑	δ (%)↓
Deep Research	48	26	29.7	21.7	41.9	31.8
AutoGPT	2	56	93.1	8.7	25.8	49.6
WebVoyager	58	40	18.4	39.1	54.8	16.7
OpenAI Research Agent	10	14	16.7	14.5	9.7	19.8
Search + TAG	60	20	50	10.1	9.7	**2.36**
DramaBot	**84**	**92**	**4.5**	**82.6**	**90.3**	4.5

DramaBot generalizes across fine-grained task types within each category. As we show in Table 5, while baseline agents exhibit large disparities in performance across different task types, DramaBot maintains consistently low variation, indicating strong generalization. Specifically, the relative difference in accuracy across all task types for DramaBot remains below 5%. Within verification tasks, the relative difference between accuracy on True and False instances for baseline agents ranges from 3.7 to 21.7 times higher than that of DramaBot. In QA tasks, the disparity between performance on questions with numeric and string answers for baseline agents is 3.7 to 11 times greater than that of DramaBot.

DramaBot maintains superior performance across all time spans. To evaluate performance over time, we assign each task a timestamp based on its publication date. Specifically, for Politifact claims, we use the original date the claim was made for the False claim and the date the corresponding claim was analyzed and posted on Politifact for the True claim; for Twitter/X Community Notes, we use the time of the original tweet for the False claim and the time when the corresponding community note was posted for the True claim; for USAFacts, we use the article publication date. We present the overall accuracy (in percentage) of different agents across task instances with varying timestamps in Figure 5. We can observe that the relative ranking of the baseline agents fluctuates over time. This indicates that DramaBench effectively captures the temporal dynamics of real-world data analysis tasks. Despite these fluctuations, DramaBot consistently outperforms all baseline agents across all time periods, demonstrating its robustness to temporal shifts and its effectiveness even as the relevance and availability of information evolve over time.

5.4 Stage-wise Strength Underpins DramaBot's End-to-end Performance

We evaluate DramaBot 's stage-wise performance by assessing its intermediate outputs, specifically the quality of the retrieved data and the generated analytic code. We summarize the quantitative results in Table 6, and demonstrate our findings in detail as follows.

Consistency across different similarity metrics supports robust comparison. We first evaluate the effectiveness of the similarity metrics by measuring their mutual agreement rate. This rate is defined as the average pairwise agreement across all metrics, where each pair determines whether the relative ranking of items is preserved. A higher mutual agreement rate indicates stronger consistency among the metrics in capturing item-level similarity.

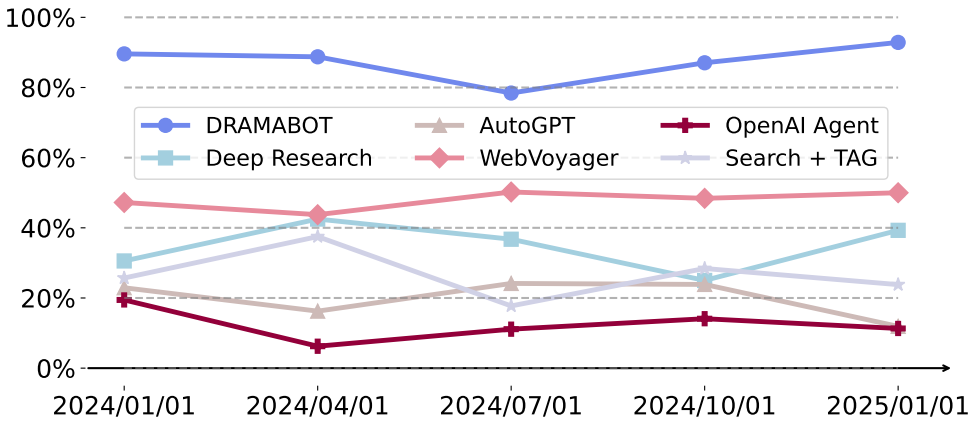


Fig. 5. The overall accuracy (%) of different agents across time. Each labeled time point marks the start of a three-month period.

Table 6. Component-level performance of different agents. The optimal values of critical metrics are highlighted with both bold and underline formatting.

(a) **Data Component**

Agents	**Data Validity (%)↑**	**Data Similarity**				
		Sim1↑ (rk↓)	Sim2↑ (rk↓)	Sim3↑ (rk↓)	Sim4↑ (rk↓)	***Avg↑ (rk↓)***
Deep Research	75.0	0.485 (4)	0.461 (4)	0.607 (4)	0.435 (4)	*0.497 (4)*
AutoGPT	26.0	0.143 (6)	0.134 (6)	0.185 (6)	0.125 (6)	*0.147 (6)*
WebVoyager	75.0	0.470 (5)	0.433 (5)	0.608 (3)	0.427 (5)	*0.484 (5)*
OpenAI Research Agent	96.0	0.628 (2)	0.571 (1)	0.793 (2)	0.530 (1)	*0.631 (2)*
Search + TAG	74.0	0.485 (3)	0.464 (5)	0.604 (2)	0.440 (3)	*0.498 (3)*
DRAMABOT	**97.5**	0.649 (1)	0.557 (2)	0.802 (1)	0.525 (2)	***0.633 (1)***

(b) **Code Component**

Agents	**Code Execution (%)↑**	**Code Similarity**				
		Sim1↑ (rk↓)	Sim2↑ (rk↓)	Sim3↑ (rk↓)	Sim4↑ (rk↓)	***Avg↑ (rk↓)***
Deep Research	61.5	0.420 (4)	0.735 (4)	0.406 (4)	0.735 (4)	*0.574 (4)*
AutoGPT	11.0	0.328 (5)	0.328 (5)	0.258 (5)	0.156 (5)	*0.250 (5)*
WebVoyager	66.5	0.667 (1)	0.739 (3)	0.673 (2)	0.740 (3)	*0.704 (3)*
OpenAI Research Agent	94.5	0.611 (3)	0.945 (1)	0.684 (1)	0.951 (1)	***0.798 (1)***
DRAMABOT	**95.0**	0.634 (2)	0.932 (2)	0.619 (3)	0.937 (2)	*0.781 (2)*

For data similarity, the mutual agreement rate is 82.2%. All four metrics consistently reflect the same overall trend: DRAMABOT and the OpenAI Research Agent rank highest with similar scores, followed by Deep Research, Search + TAG, and WebVoyager, with AutoGPT consistently ranked lowest. For code similarity, the mutual agreement rate is 83.3%, with the metrics agreeing

on the relative rankings: the OpenAI Research Agent produces the most similar code, followed by DramaBot, WebVoyager, Deep Research, and AutoGPT.

Strong correlation across Drama stages highlights the benefit of unification. The quality of the collected data and the generated code are highly correlated, with a Pearson coefficient [53] of $\rho = 0.97$ between the average data similarity and code similarity. This highlights the significance of Drama in unifying the pipeline to ensure consistency and effectiveness across both data retrieval and code generation.

DramaBot retrieves data of high quality. We observe that DramaBot retrieves data with the highest validity rate, reaching up to 3.8× of that of AutoGPT. It also achieves the highest average data similarity, up to 4.3× of that of AutoGPT. These results demonstrate the effectiveness of the data retriever in both the data collection and data transformation stages in Drama.

DramaBot's data extraction achieves 90% accuracy. We rerun DramaBot on the 27 failure cases by providing direct access to the collected data, either the original websites where the data is embedded or the readily downloaded raw files. Since successful end-to-end task completion indicates accurate data extraction, DramaBot achieves 90% extraction accuracy across all DramaBench tasks, with 92% for verification tasks and 88% for QA tasks. This demonstrates the effectiveness of DramaBot's strategy.

DramaBot demonstrates strong analytic reasoning capabilities. We observe that DramaBot achieves the highest code execution rate, reaching up to 8.6× that of AutoGPT. Although DramaBot ranks second in code similarity, the difference from the highest score is less than 0.02.

5.5 Web Browser and Web Augmenter Coordinate for Data Collection

We proceed to analyze the behavior of the data retriever to understand its effectiveness in detail. We first examine how it coordinates the web browser and the web augmenter, which together execute the data collection stage in Drama, and then evaluate their individual contributions.

Table 7. The percentage of task instances where the data is collected by different subagents of DramaBot and their corresponding accuracy.

Sub-agent	Workload Share (%)	Accuracy (%)
Web Browser	63.5	91.2
Web Augmenter	36.5	68.0

Data retriever successfully coordinates the web browser and web augmenter for balanced workload and accuracy. We can see from Table 7 that in 63.5% of the task instances, the web browser collects the data, while the remaining 36.5% are handled by the web augmenter. This demonstrates that the coordinating capability of the data retriever results in a reasonable division of labor that aligns with each sub-agent's accuracy. Specifically, the web browser, which explores data at a finer level of granularity and is more effective at handling DramaBench tasks, takes on a larger share of the workload to ensure strong overall system performance.

We further conduct ablation analysis by removing the web browser and the web augmenter individually, and evaluate the system on all DramaBench tasks.

Both web augmenter and web browser enhance DramaBot's performance. As we compare the performance of DramaBot with different sub-agents removed in Table 8 to that of the original

Table 8. Accuracy (%) of DRAMABOT on different DRAMABENCH tasks with different components removed.

Setting	Overall	Verification	QA	Workload Share (%) in Table 7
Web Browser only	59	57	61	60.7
Web Augmenter only	53	59	47	52.1

DRAMABOT in Table 3, we observe that both subagents play a critical role in ensuring the system's overall effectiveness. Specifically, removing the web browser results in a 33.5 percentage point drop in performance, while removing the web augmenter leads to a 27.5 point drop.

Both web augmenter and web browser demonstrate strong standalone effectiveness. We observe in Table 8 that although the accuracy of DRAMABOT decreases when individual sub-agents are removed, both the web augmenter and the web browser alone still deliver strong performance in data retrieval and coordinate well with the remaining components, outperforming all baseline agents as we present in Table 3. Specifically, WebVoyager, which serves as the foundation for DRAMABOT's web browser, has an overall accuracy that is 12.5 percentage points lower than the version of DRAMABOT using only the web browser. Similarly, the OpenAI Research Agent, which also relies on the OpenAI web search tool for data collection, achieves less than 1/4 the accuracy of the version of DRAMABOT using only the web augmenter. These results highlight two key findings: (1) the overall DRAMABOT architecture effectively and reliably orchestrates its components to maintain high performance, even when certain modules are removed or operating suboptimally; and (2) the reasoning mechanisms adapted to the individual sub-agents are robust and capable of maintaining strong task performance in isolation.

The interaction between web augmenter and web browser improves each other's performance. As we compare the performance of DRAMABOT with a single sub-agent in Table 8 to the case where the full DRAMABOT executes and collects data using that same sub-agent in Table 7, we observe that the interaction between the web browser and the web augmenter enhances each other's performance. Specifically, when the web browser collects data in the presence of the web augmenter, its accuracy improves by 30.5 percentage points compared to operating alone. Similarly, when the web augmenter collects data with support from the web browser, its accuracy increases by 15 percentage points. This indicates that the integration of the web browser and the web augmenter goes beyond functioning as a simple oracle. Instead, it reflects a synergistic relationship among DRAMABOT 's sub-agents, leading to mutually reinforced improvements in data retrieval quality and reasoning capabilities.

6 Related Work

We review related literature in the following three areas.

Fact-checking models and benchmarks. Existing fact-checking benchmarks [28, 38, 62, 63, 73] primarily focus on text retrieval or statement-level summarization, without requiring precise answers grounded in analytic reasoning. For example, ClaimVerify [38] verifies claims such as "What are the latest discoveries from the James Webb Space Telescope?" using binary factual consistency annotations for each cite-worthy sentence. This setup centers on information extraction rather than performing grounded analysis or calculations. These limitations highlight the gap in existing benchmarks, which are insufficient for evaluating the full DRAMA pipeline, and underscores the significance of DRAMABENCH in advancing benchmarks for analytic, data-grounded fact verification.

Existing fact-checking models fall short in important ways for end-to-end data discoveries. Models that rely on web search [11, 12, 29] often lack the ability to process long and complex information locally, such as reasoning over full documents. On the other hand, models that operate over local documents [13, 18, 20, 31, 32, 41, 42, 62, 74], including those based on the RAG paradigm [7, 35], assume access to a local, static, and predefined set of data, and are therefore unable to retrieve information from open-domain sources. These limitations highlight the need for a unified approach that integrates both data collection and data transformation for more effective fact verification. This is precisely the problem that Drama is designed to solve, and DramaBot demonstrates its capability to carry out this unified pipeline in practice.

Dynamic and up-to-date knowledgebases. Knowledge bases have significantly enhanced the capabilities of LLMs in a variety of knowledge-intensive tasks [72]. Although there has been substantial progress in constructing up-to-date knowledge bases [30, 40] to keep models aligned with recent information and to evaluate their ability to identify and extract time-sensitive knowledge, these updates are performed manually. The models themselves do not actively retrieve new information. For example, RealTime QA [30] provides a regularly updated evaluation platform, with updates released weekly in the current version. However, the retrieval process is manually configured, limited to a fixed dataset, and not based on open-domain access. Most importantly, from the perspective of the model, the data is treated as local, static, and predefined. Streaming QA represents a similar case, where data from all dates is made available at once rather than continuously updated [40]. These limitations underscore the need for systems that support dynamic, open-domain information retrieval and reasoning. This is precisely the challenge that Drama is designed to address.

Intelligence data systems. The development of large language models (LLMs) has enabled the automation of many tedious aspects of data analysis, ranging from data preprocessing [5, 24, 25] to semantic querying [2, 10, 17, 26, 37, 52, 56, 81]. However, all of these systems operate over a static collection of data. With the exception of TAG [2], which integrates retrieval-augmented generation with NL2SQL to construct a query-driven database for more efficient information retrieval and execution, existing systems assume that a clean and well-formatted database is already available at query time. This strong assumption limits their applicability to real-world tasks, where data must often be collected, transformed, and queried dynamically. In contrast, Drama addresses this gap by enabling an end-to-end workflow that combines open-domain data collection with flexible, query-driven analysis.

7 Conclusion

In this paper, we propose Drama, an end-to-end paradigm that automates data science workflows by unifying data collection, transformation, and analysis. We further collect DramaBench, a benchmark consisting of 200 task instances that require retrieving and analyzing data from open domains. Alongside DramaBench, we develop DramaBot, a multi-agent system composed of a data retriever and a data analyzer. By coordinating the execution of sub-agents and optimizing the reasoning mechanisms, DramaBot achieves 86.5% accuracy on DramaBench at an average cost of only $0.05 per task, significantly outperforming all five state-of-the-art baseline agents.

Acknowledgments

We thank Jeonghwan Kim for his valuable insights on RAG systems; Qiusi Zhan, Yuxuan Zhu, and Tengjun Jin for their helpful advice on paper writing; and Yurong Liu and Zishan Su for their generous support during the revision process.

References

[1] 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. doi:10.1109/TVCG.2018.2865040

[2] Asim Biswal, Liana Patel, Siddarth 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

[3] Anjanava Biswas and Wrick Talukdar. 2024. Robustness of Structured Data Extraction from In-plane Rotated Documents using Multi-Modal Large Language Models (LLM). arXiv:2406.10295 [cs.CL] https://arxiv.org/abs/2406.10295

[4] 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, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] https://arxiv.org/abs/2005.14165

[5] Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Yuchen Mao, Wenjing Hu, Tianbao Xie, Hongshen Xu, Danyang Zhang, Sida Wang, Ruoxi Sun, Pengcheng Yin, Caiming Xiong, Ansong Ni, Qian Liu, Victor Zhong, Lu Chen, Kai Yu, and Tao Yu. 2024. Spider2-V: How Far Are Multimodal Agents From Automating Data Science and Engineering Workflows? arXiv:2407.10956 [cs.AI] https://arxiv.org/abs/2407.10956

[6] Central Statistics Office Ireland. 2024. *Vital Statistics Yearly Summary 2023.* https://www.cso.ie/en/releasesandpublications/ep/p-vsys/vitalstatisticsyearlysummary2023/

[7] Jiangui 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. doi:10.1145/3583780.3614821

[8] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG] https://arxiv.org/abs/2107.03374

[9] Community Notes. 2025. Download Data — Community Notes Guide. https://communitynotes.x.com/guide/en/under-the-hood/download-data. https://communitynotes.x.com/guide/en/under-the-hood/download-data

[10] Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 281–296. doi:10.18653/v1/2020.findings-emnlp.27

[11] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web* (New York, NY, USA) *(WWW '04)*. Association for Computing Machinery, New York, NY, USA, 100–110. doi:10.1145/988672.988687

[12] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: the second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One* (Barcelona, Catalonia, Spain) *(IJCAI'11)*. AAAI Press, 3–10.

[13] Alexander Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. QAFactEval: Improved QA-Based Factual Consistency Evaluation for Summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 2587–2601. doi:10.18653/v1/2022.naacl-main.187

[14] Federation for American Immigration Reform (FAIR). 2023. 2023 Illegal Alien Population Estimate. https://www.fairus.org/sites/default/files/2023-06/2023%20Illegal%20Alien%20Population%20Estimate_2.pdf.

[15] Federation for American Immigration Reform (FAIR). 2025. Federation for American Immigration Reform. https://www.fairus.org.

[16] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages.

arXiv:2002.08155 [cs.CL] https://arxiv.org/abs/2002.08155

[17] 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. *CoRR* abs/2308.15363 (2023).

[18] Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023. RARR: Researching and Revising What Language Models Say, Using Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 16477–16508. doi:10.18653/v1/2023.acl-long.910

[19] Google. 2024. Gemini Deep Research. https://gemini.google/overview/deep-research/.

[20] Tanya Goyal and Greg Durrett. 2021. Annotating and Modeling Fine-grained Factuality in Summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, Online, 1449–1462. doi:10.18653/v1/2021.naacl-main.114

[21] Significant Gravitas. 2023. Auto-GPT: An Autonomous GPT-4 Experiment. https://github.com/Significant-Gravitas/AutoGPT GitHub repository.

[22] Orit Hazzan and Koby Mike. 2023. *The Data Science Workflow*. Springer International Publishing, Cham, 151–163. doi:10.1007/978-3-031-24758-3_10

[23] 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:2401.13919 [cs.CL] https://arxiv.org/abs/2401.13919

[24] Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Chenxing Wei, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Xiangru Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zhibin Gou, Zongze Xu, and Chenglin Wu. 2024. Data Interpreter: An LLM Agent For Data Science. arXiv:2402.18679 [cs.AI] https://arxiv.org/abs/2402.18679

[25] 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. doi:10.14778/3705829.3705843

[26] Chuxuan Hu, Liyun Zhang, Yeji Lim, Aum Wadhwani, Austin Peters, and Daniel Kang. 2025. REPRO-Bench: Can Agentic AI Systems Assess the Reproducibility of Social Science Research?. 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, 23616–23626. https://aclanthology.org/2025.findings-acl.1210/

[27] Indian Tech & Infra. 2024. Uttar Pradesh is now the second-largest economy in the country, competing to become India's top economy: CM Yogi Adityanath. https://x.com/IndianTechGuide/status/1850054756921925792.

[28] Ryo Kamoi, Tanya Goyal, Juan Diego Rodriguez, and Greg Durrett. 2023. WiCE: Real-World Entailment for Claims in Wikipedia. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 7561–7583. doi:10.18653/v1/2023.emnlp-main.470

[29] Serafina Kamp, Morteza Fayazi, Zineb Benameur-El, Shuyan Yu, and Ronald Dreslinski. 2023. Open Information Extraction: A Review of Baseline Techniques, Approaches, and Applications. arXiv:2310.11644 [cs.IR] https://arxiv.org/abs/2310.11644

[30] Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2024. RealTime QA: What's the Answer Right Now? arXiv:2207.13332 [cs.CL] https://arxiv.org/abs/2207.13332

[31] Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the Factual Consistency of Abstractive Text Summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 9332–9346. doi:10.18653/v1/2020.emnlp-main.750

[32] Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. 2022. SummaC: Re-Visiting NLI-based Models for Inconsistency Detection in Summarization. *Transactions of the Association for Computational Linguistics* 10 (02 2022), 163–177. arXiv:https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00453/1987014/tacl_a_00453.pdf doi:10.1162/tacl_a_00453

[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] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2025. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. arXiv:2411.07763 [cs.CL] https://arxiv.org/abs/2411.07763

[35] 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

[36] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi 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).

[37] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baille Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, and Gerardo Vitagliano. 2024. A Declarative System for Optimizing AI Workloads. arXiv:2405.14696 [cs.CL] https://arxiv.org/abs/2405.14696

[38] Nelson Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating Verifiability in Generative Search Engines. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 7001–7025. doi:10.18653/v1/2023.findings-emnlp.467

[39] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173. doi:10.1162/tacl_a_00638

[40] Adam Liška, Tomáš Kočiský, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, Cyprien de Masson d'Autume, Tim Scholtes, Manzil Zaheer, Susannah Young, Ellen Gilsenan-McMahon, Sophia Austin, Phil Blunsom, and Angeliki Lazaridou. 2022. StreamingQA: A Benchmark for Adaptation to New Knowledge over Time in Question Answering Models. arXiv:2205.11388 [cs.CL] https://arxiv.org/abs/2205.11388

[41] Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. 2024. ExpertQA: Expert-Curated Questions and Attributed Answers. arXiv:2309.07852 [cs.CL] https://arxiv.org/abs/2309.07852

[42] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 12076–12100. doi:10.18653/v1/2023.emnlp-main.741

[43] National Park Service. 2023. *Visitor Spending Effects: Economic Contributions of National Park Visitor Spending.* https://www.nps.gov/nature/customcf/NPS_Data_Visualization/docs/NPS_2023_Visitor_Spending_Effects.pdf

[44] National Park Service. 2025. Nature - U.S. National Park Service. https://www.nps.gov/nature/. https://www.nps.gov/nature/

[45] OpenAI. 2024. GPT-4o mini. https://platform.openai.com/docs/models/gpt-4o-mini.

[46] OpenAI. 2024. GPT-4o Overview. https://platform.openai.com/docs/models/gpt-4o. Accessed: 2025-04-11.

[47] OpenAI. 2024. Introducing GPT-4o. https://openai.com/index/hello-gpt-4o/. Accessed: 2025-08-03.

[48] OpenAI. 2024. O3 Mini Model Documentation. https://platform.openai.com/docs/models/o3-mini.

[49] OpenAI. 2024. OpenAI Agents Python: Research Bot Example. https://github.com/openai/openai-agents-python/tree/main/examples/research_bot. Accessed: 2025-04-08.

[50] OpenAI. 2024. OpenAI Embeddings Guide. https://platform.openai.com/docs/guides/embeddings.

[51] OpenAI. 2024. Tools and Web Search | OpenAI Platform. https://platform.openai.com/docs/guides/tools-web-search?api-mode=chat Accessed: 2025-04-08.

[52] 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

[53] 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.

[54] Python Software Foundation. 2024. *ast — Abstract Syntax Trees.* Python Software Foundation. https://docs.python.org/3/library/ast.html

[55] Selenium Project. 2025. Selenium: Web Browser Automation. https://www.selenium.dev/. https://www.selenium.dev/ Accessed: 2025-08-04.

[56] 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

[57] Statista. 2024. *India: PM Air Pollution Exposure.* https://www.statista.com/statistics/935636/india-pm-air-pollution-exposure/ Accessed: 2025-04-14.

[58] Statista. 2025. The U.S. Minimum Wage by State. https://www.statista.com/statistics/238997/minimum-wage-by-us-state/.
[59] Statistics Times. 2025. List of Indian States and Union Territories by GDP. https://statisticstimes.com/economy/india/indian-states-gdp.php.
[60] 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. doi:10.1145/3360646
[61] Bashir Tahir. 2024. Open Deep Research. https://github.com/btahir/open-deep-research. Accessed: 2025-04-08.
[62] Liyan Tang, Philippe Laban, and Greg Durrett. 2024. MiniCheck: Efficient Fact-Checking of LLMs on Grounding Documents. arXiv:2404.10774 [cs.CL] https://arxiv.org/abs/2404.10774
[63] Liyan Tang, Igor Shalyminov, Amy Wing mei Wong, Jon Burnsky, Jake W. Vincent, Yu'an Yang, Siffi Singh, Song Feng, Hwanjun Song, Hang Su, Lijia Sun, Yi Zhang, Saab Mansour, and Kathleen McKeown. 2024. TofuEval: Evaluating Hallucinations of LLMs on Topic-Focused Dialogue Summarization. arXiv:2402.13249 [cs.CL] https://arxiv.org/abs/2402.13249
[64] Maria Ramirez Uribe. 2024. There aren't 20 million to 30 million immigrants in the U.S. illegally, as Sen. Marco Rubio claimed. https://www.politifact.com/factchecks/2024/jun/11/marco-rubio/there-arent-20-million-to-30-million-immigrants-in/.
[65] U.S. Department of Housing and Urban Development. 2024. *2024 AHAR: Part 1 - PIT Estimates of Homelessness in the U.S.* https://www.huduser.gov/portal/datasets/ahar/2024-ahar-part-1-pit-estimates-of-homelessness-in-the-us.html
[66] USAFacts. 2024. *How do national parks affect the economy?* https://usafacts.org/articles/how-do-national-parks-affect-the-economy/
[67] USAFacts. 2024. *How much damage do wildfires do in the US?* https://usafacts.org/articles/how-much-damage-do-wildfires-do-in-the-us/ Accessed: 2025-04-14.
[68] USAFacts. 2024. *Which states have the highest and lowest rates of homelessness?* https://usafacts.org/articles/which-states-have-the-highest-and-lowest-rates-of-homelessness/
[69] USAFacts. 2025. Minimum wage in America: How many people are earning $7.25 an hour? https://usafacts.org/articles/minimum-wage-america-how-many-people-are-earning-725-hour/.
[70] USAFacts. 2025. USAFacts: Nonpartisan data. Driven by facts. https://usafacts.org/. https://usafacts.org/
[71] USAFacts. 2025. USAFacts: Nonpartisan Government Data. https://usafacts.org/. https://usafacts.org/ Accessed: 2025-08-04.
[72] Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023. KnowledGPT: Enhancing Large Language Models with Retrieval and Storage Access on Knowledge Bases. arXiv:2308.11761 [cs.CL] https://arxiv.org/abs/2308.11761
[73] Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, Isabelle Augenstein, Iryna Gurevych, and Preslav Nakov. 2024. Factcheck-Bench: Fine-Grained Evaluation Benchmark for Automatic Fact-checkers. arXiv:2311.09000 [cs.CL] https://arxiv.org/abs/2311.09000
[74] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large Language Models are Better Reasoners with Self-Verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 2550–2575. doi:10.18653/v1/2023.findings-emnlp.167
[75] World Population Review. 2024. *Education Rankings by Country 2024.* https://worldpopulationreview.com/country-rankings/education-rankings-by-country Accessed: 2025-04-14.
[76] X (formerly Twitter). 2024. *Community Note clarifying claim on birth rate in Ireland.* https://x.com/i/birdwatch/n/1821558069928620304 Accessed: 2025-04-14.
[77] X (formerly Twitter). 2024. *Post on X referencing change in number of births in Ireland (2022–2023).* https://x.com/i/web/status/1821510949771059296
[78] X (formerly Twitter). 2024. *Post referencing global education rankings.* https://x.com/i/web/status/1769443227902341380 Accessed: 2025-04-14.
[79] X (formerly Twitter). 2024. *Post referencing India's cigarette consumption statistic.* https://x.com/i/web/status/1858383505501098277 Accessed: 2025-04-14.
[80] Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions. arXiv:2306.02224 [cs.AI] https://arxiv.org/abs/2306.02224
[81] Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. 2024. Data-Copilot: Bridging Billions of Data and Humans with Autonomous Workflow. arXiv:2306.07209 [cs.CL] https://arxiv.org/abs/2306.07209

Received April 2025; revised July 2025; accepted August 2025