# Graphy'our Data: Towards End-to-End Modeling, Exploring and Generating Report from Raw Data

Longbin Lai
longbin.lailb@alibaba-inc.com
Alibaba Group
Hangzhou, China

Changwei Luo
pomelo.lcw@alibaba-inc.com
Alibaba Group
Hangzhou, China

Yunkai Lou
louyunkai.lyk@alibaba-inc.com
Alibaba Group
Hangzhou, China

Mingchen Ju
mingchen.ju@student.unsw.edu.au
University of New South Wales
Sydney, Australia

Zhengyi Yang
zhengyi.yang@unsw.edu.au
University of New South Wales
Sydney, Australia

## Abstract

While Large Language Models (LLMs) excel at single-document queries and conversational workflows, they struggle with progressively exploring, analyzing, and synthesizing large unstructured document sets, such as in literature surveys. We address this challenge – termed Progressive Document Investigation – by introducing Graphy, an end-to-end platform that automates data modeling, exploration and high-quality report generation in a user-friendly manner. Graphy comprises an offline Scrapper that transforms raw documents into a structured graph, and an online Surveyor that enables iterative exploration and LLM-driven report generation. We showcase a pre-scrapped graph of over 50,000 papers, demonstrating how Graphy facilitates the literature-survey scenario, with video available at https://youtu.be/uM4nzkAdGlM.

## CCS Concepts

• **Information systems** → **Hierarchical data models**; **Content analysis and feature selection**.

## Keywords

Graph Model, Large Language Model, Document Analysis

**ACM Reference Format:**
Longbin Lai, Changwei Luo, Yunkai Lou, Mingchen Ju, and Zhengyi Yang. 2025. Graphy'our Data: Towards End-to-End Modeling, Exploring and Generating Report from Raw Data. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725106

## 1 Introduction

We study real-world investigative tasks that require iterative exploration and synthesis of large set of documents. We refer to this challenge as **Progressive Document Investigation** (PDI), an iterative process of identifying a topic, refining a relevant dataset, and ultimately generating high-quality reports. A motivating example of PDI is the literature survey process in academic research. Researchers start with a few seed papers, and conduct iterative rounds of investigation: skimming key elements (e.g., "abstract", "challenges", "solutions"), following references to additional papers, and expanding the set of relevant works. They then synthesize their findings into a structured survey report – often by grouping papers with shared characteristics (e.g., addressing similar challenges).

The advent of Large language models (LLMs) [2] have shown some promise in addressing PDI, especially when combined with techniques like Retrieval-Augmented Generation (RAG) [9] and autonomous agents [8]. While these methods excel at single-document queries and conversational workflows, they still fall short in PDI. RAG-based solutions often struggle to maintain consistency and organization when applied to large-scale, multi-step explorations. Existing agent systems, on the other hand, risk error propagation across extensive pipelines. Moreover, both methods typically provide limited support for iterative user oversight and curation to ensure accuracy and control.

To address these gaps, we propose Graphy, an end-to-end platform that streamlines the PDI workflow. We adopt the property graph model for the need of iterative exploration in PDI. Inspired from business intelligence (BI) systems [7], we introduce Fact and Dimension nodes, analogous to Fact and Dimension tables in BI. Here, Fact nodes represent the primary entities of interest, while Dimension nodes capture supplementary information. In a literature-survey context, each paper functions as a Fact node, and its extracted contents, such as "challenges", and "solutions", serve as Dimension nodes. This work centers on the literature-survey scenario, but Graphy is broadly applicable, as will be briefly discussed in Section 4. Fig. 1 provides an overview of Graphy, which consists of an offline Scrapper and an online Surveyor.

**Offline Scrapper**. The Scrapper allows users to implement the Inspection abstraction to direct the extraction of specific Dimensions from each document, often leveraging LLMs. This step transforms an unstructured document into a structured Fact node linked to predefined Dimension nodes. It simulates how a human researcher would skim a document, pinpointing aspects such as abstract, challenges, and solutions. Additionally, a Navigation abstraction defines how Fact nodes are connected, enabling the retrieval


*SIGMOD-Companion '25, Berlin, Germany*

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725106

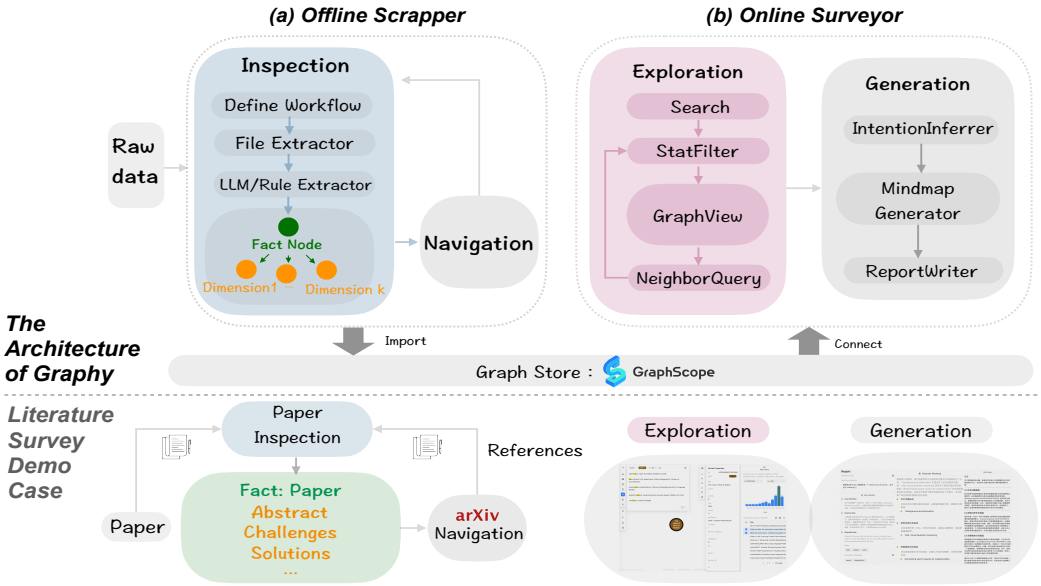


**Figure 1: The design and demo case of literature survey of Graphy.**

of related items for progressive exploration. For instance, an arxiv Navigation automatically fetches and downloads research papers from arxiv [1], if available.

Because both the extracted data from the Inspection and the linked data from the Navigation are relatively stable, we run the Scrapper offline. Upon completion, it produces a graph of Fact nodes, Dimension nodes, and their interconnecting edges, which can be imported into a graph database (e.g., GraphScope [6], kuzu [4]).

**Online Surveyor**. Designing a user-friendly Surveyor on top of graph databases poses two key challenges. First, unlike SQL, graph query languages are less familiar to users. Second, graph exploration can become unwieldy, particularly with "supernodes," which have extremely large numbers of connections. We address these challenges with the Exploration, which is the main interface for navigating the graph and selecting papers of interest. As shown in Fig. 1, it offers a convenient `Search` module to initiate exploration. Users can iteratively move from one set of nodes to their neighbors, with graph queries (e.g. `NeighborQuery`) seamlessly integrated into UI interfaces. To avoid overwhelming users, Exploration employ `StatFilter` that provides histograms and top-k selectors, allowing users filter out neighbors of interests.

Eventually, users can proceed to the Generation module, which leverages LLMs for crafting reports from the papers selected in the Exploration. Users specify the report's focus, and the Generation attempts to infer necessary attributes and relevant Dimension nodes (e.g., "challenges", "solutions") for drafting a mind map. After reviewing, the system produces a coherent, structured report that mimics a human researcher's synthesis process. The final document can be exported in format like TeX to facilitate academic writing.

In this paper, we demonstrate how Graphy can streamline the literature-survey process. Specifically:

- **Data Extraction and Linking:** With a predefined workflow, we demonstrate how the Scrapper employs the Inspection to extract Fact and Dimension nodes from research papers, and how the Navigation automatically expands from a set of seed papers to their cited references.
- **Paper Exploration:** Using a pre-scrapped graph containing approximately 50,000 papers, 250,000 Dimension nodes, and 160,000 references among the papers, we demonstrate how users can utilize the Exploration interface to progressively search for papers of interest, simulating the process of literature survey.
- **Report Generation:** We demonstrate how the Generation collects essential information from the selected papers and creates mind maps in line with users' intentions. We then showcase its capability to transform these mind maps into a well-structured report, which users can download in formats such as TeX.

We have open-sourced both the Graphy codebase and the pre-scrapped research graph [3].

## 2 Architecture

This section introduces the architecture of Graphy, which comprises an Offline Scrapper and an Online Surveyor.

### 2.1 Offline Scrapper

**Inspection**. Given a paper document as input, the Inspection processes it to produce a graph structure. The paper itself forms the Fact node, while its Dimension nodes are extracted following the instructions of a Directed Acyclic Graph (DAG). Each subnode's definition aligns with the user's requirements. For simple dimensions (e.g., an "abstract"), users can employ rule-based methods such as regular expressions. For more advanced tasks, the system supports individually configured LLM subnodes, allowing users to balance cost and performance. For instance, simpler processing can rely on smaller locally deployed models, whereas more complex extraction may involve sophisticated cloud-based models [2]. These LLM-based subnodes build on a common RAG workflow that

chunks PDF text, stores it in a vector database, and then retrieves only the most relevant chunks based on user-defined queries.

A snippet of the `PaperInspection` DAG in Fig. 1 is shown below. The subnode `Abstract` is rule-based for extracting the paper's "abstract", while two LLM-based subnodes form a chain: `Challenges` uses a locally deployed model (prefixed with "ollama/") to identify challenges, and `Solutions` leverages a cloud-based model to extract solutions. Such chain formation allows the `Solutions` subnode to leverage the context provided by the `Challenges` subnode.

```
"nodes": [ ...,
    {
      "name": "Abstract",
      "extract_from": { ... }, # the rule of extracting abstract
      "output_schema": { single_typed: ... } # the output formats
    },
    {
      "name": "Challenges",
      "model" : { "name": "ollama/qwen2.5:7b", ... },
      "query": "Please summarize the challenges in this paper",
      "output_schema": { array_typed: ... } # the output formats
    },
    {
      "name": "Solutions",
      "model" : { "name": "qwen-plus", ... },
      "query": "Please summarize the solutions in this paper",
      "output_schema": { array_typed: ... }
    }, ...,
],
"edges": [ ...,
  {"source": "Challenges", "target": "Solutions" }
]
```

We have manually defined the Inspection DAG for demonstration purpose. However, it's straightforward to leverage LLMs to generate such DAGs from human-curated examples. Additionally, we focus on extracting text from documents and plan to extend this to process images, tables, and other non-text elements.

**Navigation**. The Navigation is responsible for establishing connections between Fact nodes, and in this case particularly, linking papers through their references. Specifically, a subnode can be deployed in the above "PaperInspection" to extract references from a paper. These references are then processed by the Navigation to fetch additional paper documents. Currently, we have implemented a Navigation to retrieve papers from arxiv [1]. For each reference, only those that can be matched and retrieved through the Navigation are retained. The corresponding documents are downloaded, and the Inspection workflow is repeated for these new papers.

**Graph Modelling**. The results of Inspection and Navigation, as shown in Fig. 1, naturally form a graph comprising Fact and Dimension nodes. Each Fact node represents a paper, while the outputs generated by subnodes in the Inspection form a set of Dimension nodes linked to their corresponding Fact node. This graph is incrementally expanded as new papers are processed. Specifically, when a new Fact node $p_2$ is added, it is linked to an existing Fact node $p_1$ if $p_2$ is retrieved from the references of $p_1$.

A notable feature of the Inspection is the customizable "output_schema" for each subnode in the DAG, which defines the schema (data fields with data types) for the resulting Dimension nodes. The output can be single-typed, such as "abstract" and "title" of the paper, which can be directly stored as attributes of the Fact node. Array-typed outputs like "challenges" and "solutions" can be stored as separate Dimension nodes, each sharing the same schema.

## 2.2 Online Surveyor

**Exploration**. Traditional graph exploration typically relies on query languages, which can require extra effort to master. We address this by embedding graph queries within interactive UI components. As shown in Fig. 2, the `Search` module in the Exploration helps users pinpoint their initial papers for exploration. Three key interactions are highlighted: "E1" searches all nodes containing the "year" attribute with a single click; "E2" displays a histogram of nodes by "year" providing a statistical overview; and "E3" filters and retrieves nodes for a specific year (e.g., 2023) by clicking the corresponding histogram bar. These user actions are seamlessly translated into Cypher queries and executed on the underlying graph database.

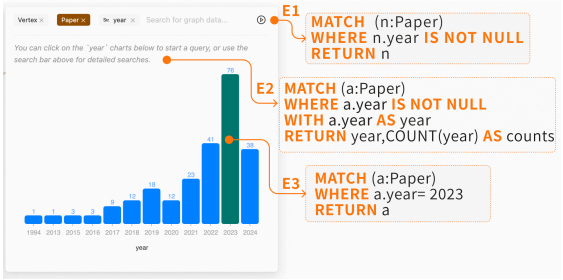


**Figure 2: The Search component of Graphy.**

Furthermore, encountering "supernodes" with exceedingly large numbers of connections can often overwhelm users and disrupt the analysis flow. To address this, we introduce a `StatFilter` module that intervenes before displaying all the neighbors. This module can present neighbors either as a histogram, allowing users to quickly overview and multi-select by groups, or as a table, where they can sort by specific attributes and choose the top-k results for further exploration. In Section 3, we provide examples showing how this approach streamlines the exploration process.

**Generation.** Once users finish selecting papers in the Exploration, they can employ the Generation to convert this explored data into structured reports. By leveraging the capabilities of LLMs, the Generation turns the network of interconnected papers on the canvas into a mind map and, ultimately, a well-organized report. This process involves three main steps: (1) **Inferring User Intentions**: Users describe their desired report in natural language, from which LLM infers which attributes and dimensions of the paper are needed. For instance, if a user asks for a related work section focusing on the paper's challenges, the LLM may determine that the "title" and "abstract" attributes and the "challenges" dimension are required. Users can review and refine these selections before proceeding. (2) **Generating Mind Maps**: Like a human expert, we prompt the LLM to organize the selected papers into a mind map based on the given dimensions, providing a high-level blueprint for the final report. To accommodate context-size limitations, we adopt an iterative approach that feeds the LLM subsets of the data at a time, gradually constructing the mind map for users to review. (3) **Writing Reports**: With the mind map in place, the LLM finalizes the literature survey by generating a cohesive report, which can then be downloaded in format like Tex to support academic writing.

# 3 Demonstrating Literature Survey

We demonstrate how Graphy applies to literature surveys, with emphasis on the online Surveyor.

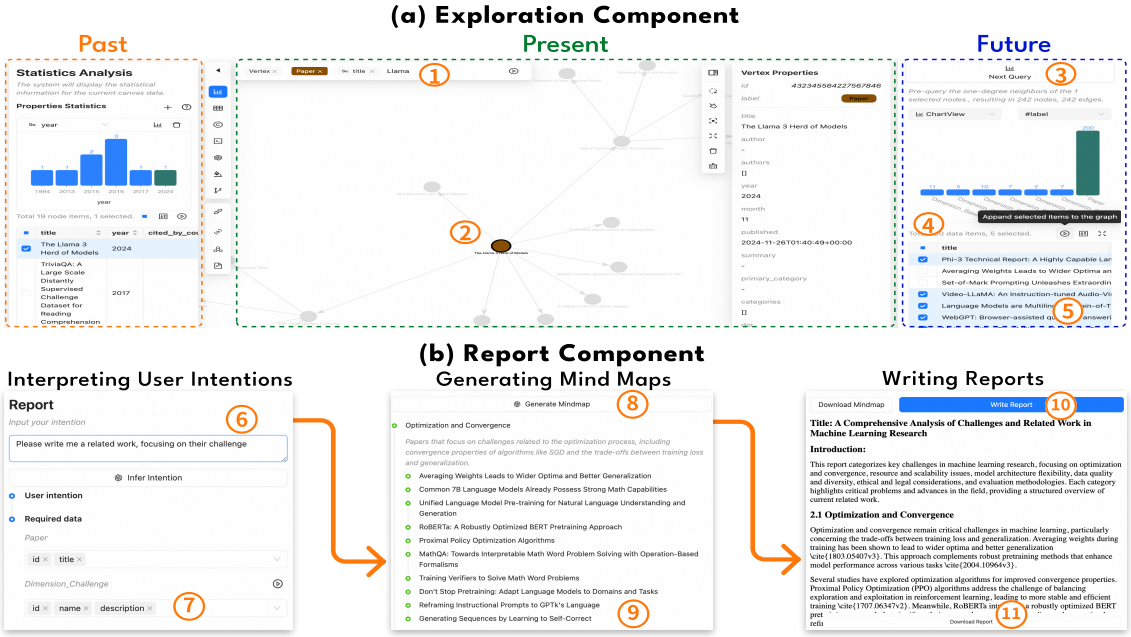


**Figure 3: The demonstration scenario of literature survey of Graphy.**

The online Surveyor, shown in Fig. 3, allows the demo attendees to explore a pre-scrapped paper network containing over 50,000 papers and 160,000 references. We first look into Fig. 3(a) that is the interface of Exploration featuring three primary canvases, metaphorically referred to as "Past", "Present", and "Future". Here, "Past" displays already explored papers, and "Present" shows the currently active papers for reviewing in detail, while "Future" highlights the references of the active papers. For exploring the papers, the attendee ① searches for seed papers whose titles contain "Llama3" using the `Search` module; ② then selects "The Llama 3 Herd of Models" and moves it to the "Present" canvas to review its details. Next, ③ the attendee explores the selected paper's references by pre-querying its neighbors. As described in Section 2.2, these neighbors are not immediately added to the canvas to avoid overwhelming the user; instead, ④ the `StatFilter` module presents a histogram or table view, allowing attendees to focus on aggregated groups or order the data and finally, ⑤ decide from the top-k papers for further exploration. By doing so, these papers are added to the "Present"canvas, while the previously active papers move to the "Past" canvas. By iteratively following this workflow, attendees can explore as many papers as needed.

In Fig. 3(b), ⑥ attendees click to input instructions for the report. Based on this input, an LLM (QWen-Plus [5] for this demo) identifies the relevant attributes and Dimension nodes needed for the report, which are ⑦ displayed for user verification and possible modification. In the example, the LLM highlights the "Challenge" node as well as the "id" and "title" attributes from the selected papers. ⑧ These data are then passed to the LLM to produce a mind map, effectively categorizing the papers according to the identified "Challenge". ⑨ Attendees can review the mind map, and ⑩ proceed to final report generation. Once completed, ⑪ attendees can download the report in TeX format with citations.

## 4 Extension to Financial Scenarios

We briefly discuss applying Graphy to two financial scenarios.

**Company Relationship Analysis**. Each company is treated as a Fact node, and the data extracted by Inspection, such as revenues, main business areas and shareholder holdings from financial reports, are represented as Dimension nodes. The Navigation component establishes inter-company relationships by leveraging the financial or supply-chain dependencies mentioned in the reports. The generated graph can be used to identify competitors, uncover hidden relationships, or assess contagion effects.

**Financial News Analysis**. Each news article serves as a Fact node, while pertinent details, such as described events and stock performance indicators, can act as Dimension nodes. The Navigation builds connections among these Fact nodes by identifying shared symbols or overlapping financial metrics. This allows analysts to track the evolution of news stories, assess their market impact, or predict future trends based on historical patterns.

## References

[1] 2024. arXiv.org e-Print archive. https://arxiv.org/.
[2] 2024. *GPT-4o, the new flagship model that can reason across audio, vision, and text in real time.* https://openai.com/index/hello-gpt-4o/
[3] 2024. Graphy. https://github.com/GraphScope/portal/tree/main/python/graphy.
[4] 2024. Kuzu. https://kuzudb.com/.
[5] 2024. *Top-performance foundation models from Alibaba Cloud.* https://www.alibabacloud.com/en/solutions/generative-ai/qwen?_p_lc=1
[6] Wenfei Fan, Tao He, Longbin Lai, and et al. 2021. GraphScope: A Unified Engine for Big Graph Processing. *Proc. VLDB Endow.* 14, 12 (jul 2021), 2879–2892.
[7] J. Gray, A. Bosworth, A. Lyaman, and H. Pirahesh. 1996. Data cube: a relational aggregation operator generalizing GROUP-BY, CROSS-TAB, and SUB-TOTALS. In *ICDE*. 152–159.
[8] Shanshan Han, Qifan Zhang, and et al. 2024. LLM Multi-Agent Systems: Challenges and Open Problems. https://arxiv.org/abs/2402.03578
[9] Patrick Lewis, Ethan Perez, and et al. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. https://arxiv.org/abs/2005.11401