# TabAgent: A Multi-Agent Table Extraction Framework for Unstructured Documents

Jingfei Wu
Beijing Institute of Technology
Beijing, China
jingfeiwu@bit.edu.cn

Chaoyuan Shen
Beijing Institute of Technology
Beijing, China
scy@bit.edu.cn

Qiyan Deng
Beijing Institute of Technology
Beijing, China
qiyandeng@bit.edu.cn

Yuping Wang
Beijing Institute of Technology
Beijing, China
wyp_cs@bit.edu.cn

Jiajun Li
Beijing Institute of Technology
Beijing, China
lijiajun@bit.edu.cn

Yuhao Deng
Beijing Institute of Technology
Beijing, China
dyh18@bit.edu.cn

Minghe Yu
Northeastern University
Shenyang, China
yuminghe@mail.neu.edu.cn

## ABSTRACT

With the increasing amount of unstructured documents in various domains, extracting structured data from such sources has become critical for efficient data management and advanced analytics. However, existing methods for structured table extraction face challenges: (1) the complexity and implicitness of semantic context and patterns in unstructured documents hinder accurate information extraction, and (2) limited adaptability to evolving user intents and requirements for strict schema alignment and structural constraints. To address these limitations, we propose `TabAgent`, a novel multi-agent collaborative framework for structured table extraction from unstructured documents. `TabAgent` integrates four specialized agents, Schema Agent, Extraction Agent, Semantic Agent, and Validation Agent with a shared memory repository to iteratively refine extraction results. By leveraging collaborative reasoning and iterative self-checking cycles, `TabAgent` enables accurate, adaptive, and robust table extraction across diverse document domains and user instructions. Extensive experiments on two datasets demonstrate that `TabAgent` consistently outperforms several baselines, including pure LLM extractors and LLM-based systems, highlighting the effectiveness of this collaborative framework. Our work represents one of the first multi-agent frameworks for structured table extraction, offering an applicable solution for real-world applications.

**PVLDB Reference Format:**
Jingfei Wu, Chaoyuan Shen, Qiyan Deng, Yuping Wang, Jiajun Li, Yuhao Deng, and Minghe Yu. TabAgent: A Multi-Agent Table Extraction Framework for Unstructured Documents. VLDB 2025 Workshop: DATAI.

Minghe Yu is the corresponding author.

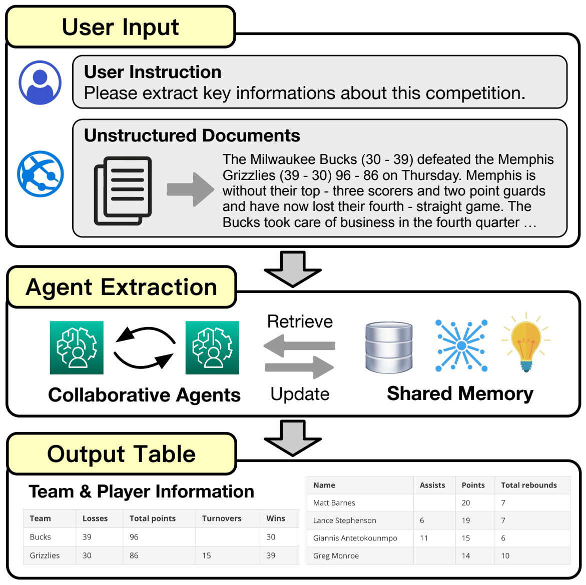
Proceedings of the VLDB Endowment. ISSN 2150-8097.

## 1 INTRODUCTION

With the rapid development of information technology, unstructured data has become increasingly common across many fields, bringing both challenges and opportunities for data processing and analysis. Large volumes of unstructured documents such as news articles, financial reports, and social media posts, are generated daily, containing valuable information that needs to be extracted accurately and efficiently for downstream analytical tasks.

Compared to unstructured formats, structured data offers significant advantages in storage, querying, and analysis. Entities with the corresponding attributes mentioned in the source documents can be extracted and organized to form a structured view. Tabular data, as a highly structured and standardized format, provides additional benefits over other forms of structured representations such as relational triples. By extracting entities and their attributes from raw text and organizing them into tables, data can be more easily integrated with databases, analytical tools, and machine learning systems. However, unstructured documents often contain scattered and mixed information, requiring conversion and efficient extraction to form structured tabular data.

In the task of extracting structured tables from unstructured documents, several key challenges arise. First, the complexity and implicitness of semantic information and patterns in unstructured documents significantly affect the accuracy and completeness of the extracted information. This requires extraction frameworks with advanced understanding and reasoning capabilities, as well as adaptability to diverse contexts, ambiguous expressions, and evolving user intents. Existing solutions can be broadly categorized into rule-based systems and supervised learning pipelines. Rule-based systems [13, 20, 24, 30] attempt to detect predefined structural linguistic patterns to extract key information from documents. Supervised learning approaches are based on ML/DL models [12, 15, 21, 22] or pre-trained language models fine-tuned on domain-specific data [6, 10, 31]. However, these methods suffer from relatively high training costs and limited adaptability to unseen data formats and diverse user intents in real-world applications.



The Milwaukee Bucks (30 - 39) defeated the Memphis Grizzlies (39 - 30) 96 - 86 on Thursday. Memphis is without their top - three scorers and two point guards and have now lost their fourth - straight game. The Bucks took care of business in the fourth quarter …

| Team | Losses | Total points | Turnovers | Wins |
|---|---|---|---|---|
| Bucks | 39 | 96 | | 30 |
| Grizzlies | 30 | 86 | 15 | 39 |

| Name | Assists | Points | Total rebounds |
|---|---|---|---|
| Matt Barnes | | 20 | 7 |
| Lance Stephenson | 6 | 19 | 7 |
| Giannis Antetokounmpo | 11 | 15 | 6 |
| Greg Monroe | | 14 | 10 |

**Figure 1: The table extraction workflow in `TabAgent`.**

The emergence of large language models (LLMs) has opened new possibilities for structured information extraction [3, 26]. However, we identify the second key challenge: transforming unstructured content into highly structured tabular data requires precise alignment with predefined or dynamically generated schemas, along with validation of structural constraints. Although LLMs excel at understanding and generating natural language, directly using them as extractors struggles with enforcing strict schema compliance and structural consistency. Some existing systems [16, 23] adopt a single-stage extraction module without sufficient output validation. The multi-agent frameworks leveraging LLMs offer better performance on complex tasks. Recent studies [7, 25] provide comprehensive and adaptive information extraction approaches through the collaboration of multiple LLM agents. However, few studies have specifically explored the design of multi-agent framework tailored for structured table extraction. Therefore, it is essential to carefully design specific modules that effectively address the unique challenges of table extraction.

To address the above challenges, we propose `TabAgent`, an LLM-based multi-agent framework for structured table extraction from unstructured documents. As in the table extraction workflow shown in Fig. 1, `TabAgent` receives the input document and user instructions, then performs table extraction through interactions among collaborative agents and their shared memory. Finally, it generates the output table as the response.

`TabAgent` comprises four agents with distinct roles: (1) The Schema Agent generates candidate table schemas by analyzing the input document and user instructions, while also referencing historical schema patterns stored in the memory repository. (2) The Extraction Agent carries out table extraction based on the generated schemas. (3) The Semantic Agent performs semantic check on the extracted table and provides revision suggestions based on historical error patterns, serving as feedback for iterative refinement with the Extraction Agent. (4) The Validation Agent ensures that the final table output satisfies strict schema and formatting constraints. In addition, the memory repository is integrated as a shared knowledge base to support the coordination among agents. This repository can be dynamically updated during execution, improving the adaptability of the framework. By combining specialized agents with the shared memory module, `TabAgent` addresses both the semantic complexity of unstructured documents and the constraints required for structured table generation, delivering an accurate, adaptive and robust solution for real-world table extraction tasks.

In short, we make the following contributions:

(1) We propose `TabAgent`, a multi-agent framework for structured table extraction from unstructured documents to achieve high accuracy, adaptability and robustness. To the best of our knowledge, this is one of the first LLM-based multi-agent frameworks in structured table extraction tasks.
(2) We introduce the self-checking cycle between table extraction and semantic validation process to iteratively refine extraction results.
(3) We evaluate `TabAgent` on two datasets and demonstrate that it achieves better table extraction quality than several baselines, including pure LLMs and LLM-based information extraction systems.

## 2 FRAMEWORK DESIGN

### 2.1 Overview

`TabAgent` is an LLM-based multi-agent framework designed for structured table extraction from unstructured documents, utilizing collaborative agents to combine schema-aware generation, iterative self-checking, and structural validation mechanisms to achieve high accuracy, generalizability and robustness in table extraction. In the task of structured table extraction, we aim to identify and extract potential tables from unstructured documents. Each table consists of entities belonging to the same class, each associated with a set of common attributes. As shown in Figure 2, `TabAgent` framework employs four specialized agents to perform extraction, iterative refinement and validation, along with the memory repository that provides auxiliary knowledge for query assistance. First, we will give a brief introduction on the key components of `TabAgent`.

**Schema Agent**. Before executing table extraction, the Schema Agent generates candidate schema for tables as extraction guidance based on the input documents by analyzing user instructions and referencing historical schema templates from related tasks stored in the memory repository. This enables the Schema Agent to adapt to varying data domains and user requirements.

**Self-checking Cycle of Extraction and Semantic Agent**. The Extraction Agent performs initial table extraction based on the generated schema, while the Semantic Agent conducts semantic validation on these tables through a self-checking cycle. In the cycle, the Semantic Agent leverages historical error patterns from the Memory Repository to refine extracted tables and provides revision suggestions, which are fed back into the Extraction Agent for iterative refinement. This self-checking cycle ensures that the extracted table maintains semantic accuracy and factual consistency with the

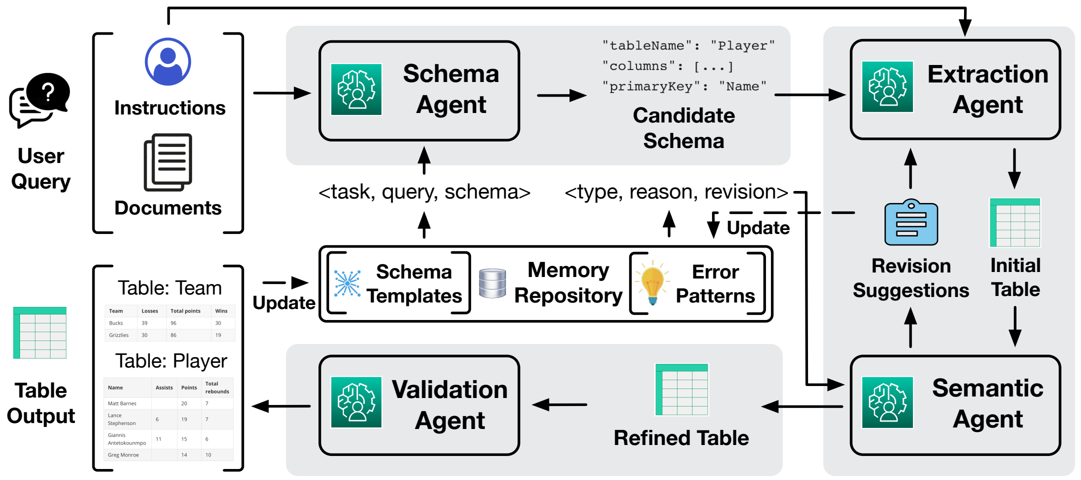


Table: Team

| Team | Losses | Total points | Wins |
|---|---|---|---|
| Bucks | 39 | 96 | 30 |
| Grizzlies | 30 | 86 | 19 |

Table: Player

| Name | Assists | Points | Total rebounds |
|---|---|---|---|
| Matt Barnes |  | 20 | 7 |
| Lance Stephenson | 6 | 19 | 7 |
| Giannis Antetokounmpo | 11 | 15 | 6 |
| Greg Monroe |  | 14 | 10 |

**Figure 2: The overall framework of `TabAgent`. `TabAgent` performs table extraction through the collaboration of specialized agents, supported by a shared memory repository.**

source documents. Additionally, these mechanisms enhance robustness of the framework against variations of document context and potential errors introduced by generative models.

**Validation Agent**. The Validation Agent obtains the refined tables resulting from the iterative refinement process and generates the final structured table output. It conducts structural validation to guarantee that the extracted table adheres to the specified schema, data types, and formatting constraints. This ensures that the output is not only semantically accurate but also ready for downstream applications such as database integration and automated analytics.

**Shared memory repository**. The memory repository acts as a centralized knowledge base, storing task-related schema templates and reflections on past errors patterns to support schema generation and iterative refinement during extraction. The repository supports dynamic update during execution, ensuring adaptation to diverse user queries.

Next, more design details of `TabAgent` framework will be presented in the following sections.

## 2.2 Schema Agent

The Schema Agent is designed to identify potential tables that can be extracted from unstructured documents and to generate the corresponding data schema for table extraction, ensuring both accuracy and adaptability across diverse user queries. The agent generates candidate table schema based on the input text, user instructions and historical knowledge. The generated schema of a table consists of the table name, a set of attributes, each defined by a name with its data type, and constraints such as primary keys or unique identifiers. The specification of table schema is critical for precise table extraction, as it ensures that extracted data aligns with the expected structure and semantic relationships.

For schema generation, the agent combines contextual reasoning with the memory repository as its knowledge base. The agent leverages task-related historical schema patterns and extraction outcomes from previous executions to better infer the implicit structure of the target table. As `TabAgent` supports user-provided extraction instructions, the agent first determines whether the user query explicitly defines the desired attributes (e.g., "Extract a table with attributes: product name, price, and sales volume."). In such cases, the agent directly maps the specified attributes to the input text using predefined rules. For general or exploratory queries without specific attributes (e.g., "Find all sales-related information in the document."), the agent consults its knowledge base to identify similar historical tasks. By analyzing these historical examples, the agent collects commonly observed attributes with their corresponding data types and infers potential schema candidates according to the user query.

In cases where the repository lacks relevant experiences, the Schema Agent updates the memory dynamically based on the final extraction results validated by other agents. The query and the associated schema from the extracted tables will be stored in memory. The framework progressively enhances its schema generation capabilities through the update mechanism, adapting to new domains and evolving user requirements.

## 2.3 Self-checking Cycle of Extraction and Semantic Agent

The self-checking cycle of the Extraction and Semantic Agent is designed to iteratively refine extracted tables through schema-guided extraction, semantic verification, and memory-augmented error resolution. Only extraction without verification may fail to ensure semantic consistency due to the inherent variability of LLMs, particularly in scenarios with ambiguous or implicit contextual patterns. Inspired by the ReAct paradigm [34], which integrates reasoning and acting in a loop and outperforms chain-of-thought (CoT) reasoning [29], the proposed self-checking cycle in `TabAgent` coordinates the Extraction and Semantic Agent within an iterative framework. The agents automatically determine whether further verification is needed, continuing the cycle until the extraction results meet the quality criteria. Through iterative refinement, the extracted data aligns with the predefined schema while upholding semantic accuracy and factual correctness.

During each iteration, the Extraction Agent generates preliminary tables based on schema definitions provided by the Schema Agent. These initial tables are then passed to the Semantic Agent, which performs a comprehensive verification process. This includes checking for semantic alignment with the original document, identifying missing or redundant attributes, and detecting inconsistencies in data representation. Detected errors are first categorized into predefined types, followed by detailed analysis to determine their root causes. Based on this analysis, the Semantic Agent provides revision suggestions that guide the next round of extraction. To enhance error detection capabilities, the Semantic Agent also retrieves historical error patterns stored in the memory repository according to the error type and task similarity. These error records, along with the revision suggestions, are injected back into the cycle as reflective feedback to enable iterative refinement.

Overall, by combining schema-guided extraction with semantic verification and memory-augmented error resolution, the framework gradually improves its performance over time, enabling accurate and robust table extraction.

## 2.4 Validation Agent

The Validation Agent performs the final validation step in the structured table extraction task. It obtains the refined table from the self-checking cycle as input and generates the final extraction result after validation, ensuring compliance with both the schema and formatting constraints. Since the table follows strictly defined schema and structural format, it imposes several specific extraction requirements, which can be categorized into two classes: (a) Structural validity and schema compliance issues, such as data type mismatches (e.g., the value "five" in an integer-type cell), duplicate rows or columns, and missing primary key values; (b) Data completeness and formatting constraints, such as incomplete records (e.g., empty rows or columns) and structural formatting errors (e.g., mismatched header-cell alignment).

To address these issues, the Validation Agent evaluates whether the extracted table satisfies the structural constraints in both rule-based logic for deterministic checks and LLM-assisted reasoning for more context-sensitive validations. These validation steps are critical for downstream tasks, as incomplete, inconsistent, or structurally flawed tables may introduce biases in data-driven models, or degrade the performance of subsequent data processing pipelines. By producing high-quality output after structural validation, the Validation Agent plays an important role in maintaining the reliability of the extracted tables.

## 2.5 Interaction among the agents and memory repository

The memory repository of `TabAgent` framework serves as a centralized knowledge base storing historical task-related schema templates and error patterns, which supports schema generation and iterative refinement process in table extraction. In view of agent-environment interaction, the memory repository functions as a repository of historical experiences, enabling agents to learn from past interactions and optimize their decision in dynamic environments, thereby enhancing the quality and robustness of table extraction process.

For the Schema Agent, the memory repository stores predefined schema templates from prior tasks. These templates are initially populated from user-defined templates, historical task logs, or external data sources. During execution, the Schema Agent queries the repository to retrieve relevant schema templates by computing embeddings of the input text and selecting the `top-k` most semantically similar extraction tasks with their schema templates. The retrieved schema templates serve as auxiliary knowledge to guide schema generation for the input documents, ensuring alignment with historical patterns and user expectations. As described in 2.2, the memory for schema templates supports dynamic update when lacking relevant experience.

Similarly, the repository also stores categorized error patterns, which are leveraged in the self-checking cycle of the Extraction and Semantic Agent. When the Semantic Agent generates revision suggestions, it retrieves associated error patterns from the repository based on the detected error type and task similarity. These historical error patterns, combined with the current revision suggestions, enable the Extraction Agent to learn from past mistakes and prevent the recurrence of similar errors in subsequent extraction cycles. The framework maintains a dynamically updated memory of error patterns, which supports the continuous evolution of its error-handling capabilities.

By integrating with multiple collaborative agents, the memory repository plays an important role in accurate and robust table extraction. It enables the overall framework to dynamically adapt to contextual variations in user queries and effectively reduce errors during the extraction process.

## 3 EXPERIMENTS

## 3.1 Basic Settings

*Datasets.* We have conducted extensive experiments to validate the effectiveness of the proposed framework, `TabAgent`, on the following two datasets.

(1) E2E [31] consists of a collection of texts of different restaurants. Each text is a description of a restaurant including its characteristics.
(2) Rotowire [31] consists of a collection of reports on basketball games. Key game information is presented in these reports, encompassing both team and player statistics. For each instance, two multi-rows tables can be extracted.

For these two datasets, we evaluate the extraction capability of `TabAgent` in scenarios where the user provides only unstructured text and general instructions without specifying the target attributes to be extracted. This setting corresponds to an open-attribute table extraction task. To initialize the memory repository, we gather documents from the relevant domain and perform table extraction. We then identify extraction errors and summarize potential refinement suggestions to form the initial set of error patterns.

*Evaluation Metrics.* In our experiments, we adopt two metrics to measure the semantic and structural accuracy of the extracted tables. First, we adopt the F1 score following [31]. For each cell in the table, we use BERTScore [37] to evaluate similarity, as LLMs may generate different but semantically equivalent descriptions for the same entity. The metric is denoted as "F1" in the following experimental results. In addition to the F1 score, we leverage the reasoning capability of LLMs to assess the accuracy of generated tables. Similarity-based metrics may inaccurately penalize cases where the output is semantically correct but inconsistent with the reference. To address this, we prompt the LLM to evaluate the accuracy, consistency, and formatting correctness between the generated and ground truth tables, and assign a corresponding evaluation score. The score ranges from 0 to 10, in which score with 0 indicates no similarity and score with 10 indicates complete equivalence between the tables. This metric is denoted as "LLM" in the following experimental result and we use GPT-4o as the LLM evaluator.

*Baselines.* We compare `TabAgent` with various baselines in the following experiments, including current information extraction systems and pure LLM extractors.

(1) Lotus [23]. We leverage the semantic operators in Lotus, including *sem_map* and *sem_extract* to extract attributes from the unstructured text. For open-attribute extraction task, we prompt the LLM to determine the attributes before extraction.
(2) Evaporate [3]. We first perform schema identification and synthesis, then filter and extract the attributes following the implementation of Evaporate-Direct.
(3) Pure LLM extractors. We leverage the reasoning capability of LLMs to extract tables directly from the documents. In our experiments, we employ LLMs of varying scales, including GPT-4o [1] as a large-scale model, and Qwen3-32B [2] as a comparatively smaller model. All models are accessed via API services.
(4) `TabAgent`. In our proposed framework `TabAgent`, we use GPT-4o model as our LLM backbone for the agents.

**Table 1: Comparison of `TabAgent` and baselines on E2E and Rotowire datasets.**

| Methods | E2E | | Rotowire/Player | | Rotowire/Team | |
|---|---|---|---|---|---|---|
| | F1 | LLM | F1 | LLM | F1 | LLM |
| GPT-4o | 55.18 | 7.55 | 65.85 | 5.02 | 51.67 | 4.30 |
| Qwen-32B | 46.46 | 6.52 | 59.51 | 4.45 | 45.32 | 3.61 |
| Lotus | 56.44 | 7.58 | 67.89 | 5.51 | 54.18 | 4.42 |
| Evaporate | 68.72 | 8.01 | 72.13 | 6.19 | 63.30 | 5.27 |
| `TabAgent` | **83.75** | **8.59** | **84.29** | **7.35** | **78.38** | **6.53** |

## 3.2 Main Results

In this section, we evaluate the proposed framework, `TabAgent`, with other information extraction baselines on the two datasets. We find that `TabAgent` outperforms all the baselines across all metrics.

Table 1 presents results on the E2E and Rotowire datasets, which correspond to the open-attribute extraction task. In these datasets, users do not specify target attributes, so agents must automatically identify key attributes that comprehensively describe the entities. Pure LLM-based solutions, which adopt a straightforward end-to-end extraction approach, perform poorly due to the lack of specialized mechanisms to ensure extraction quality. Although system like Lotus incorporates carefully designed extraction operators, it shows limited improvement over pure LLMs, primarily due to weak schema identification, particularly in settings where target attributes are not predefined. Evaporate improves upon this by introducing schema identification mechanism before extraction. However, it lacks robust mechanisms for verifying semantic and structural constraints, which limits its performance in complex scenarios like Rotowire, where the number of tables and entities varies.

Overall, `TabAgent` achieves the best performance among all baselines and shows significant improvements in open-attribute extraction tasks. Its collaborative agents are respectively responsible for schema generation, semantic refinement based on past errors, and structural validation. This modular design ensures both the accuracy of the extracted tables and adaptability to diverse user tasks.

## 3.3 Ablation Study

In this section, we evaluate the effectiveness of the specialized agents in `TabAgent` through an ablation study. The study includes four settings: the full `TabAgent` and three variants without the Schema Agent (-w/o Schema), Semantic Agent (-w/o Semantic), and Validation Agent (-w/o Validation). The Extraction Agent, which handles core extraction tasks, is not ablated. We use the Rotowire dataset for evaluation, where each document is a report summarizing a basketball game and includes two associated tables. Each table contains multiple entities along with a set of attributes and the user instruction does not specify target attributes. This complexity highlights the difficulties involved in table extraction task by LLMs.

As shown in Table 2, each agent plays a critical role in `TabAgent`'s overall performance. Removing the Schema Agent causes the largest performance drop, highlighting its role in predicting candidate schemas based on historical tasks. Accurate schema identification is essential for capturing fine-grained details and structuring table content. On the other hand, the absence of the Semantic and Validation Agents also leads to performance degradation. The Semantic Agent captures contextual and semantic relationships between table entries and input documents, offering revision suggestions for iterative refinement. The Validation Agent enforces schema and formatting constraints, as reflected by the performance drop when omitted. Due to the strong reasoning and generation capabilities of LLMs, removing these two agents has a relatively smaller impact, as the model can still produce reasonably accurate outputs.

Overall, `TabAgent` consistently outperforms its ablated variants, validating the effectiveness of collaborative agents in table extraction tasks.

## 3.4 Parameter Tuning Study

In this section, we evaluate the impact of parameter `k`, which controls the number of memory samples used by `TabAgent` during schema generation and table refinement. These samples from the memory include task-related schema templates and past error patterns. Specifically, `TabAgent` retrieves the `top-k` most relevant samples from the memory repository, including schema templates and semantic error patterns. We vary `k` from 1 to 7 and assess its effect across the datasets.

Table 2: **Ablation study results for the modules in TabAgent.**

| Methods | Rotowire/Player | | Rotowire/Team | |
|---|---|---|---|---|
| | F1 | LLM | F1 | LLM |
| TabAgent -w/o Schema | 71.22 | 6.22 | 60.53 | 4.99 |
| TabAgent -w/o Semantic | 80.67 | 7.07 | 73.24 | 6.13 |
| TabAgent -w/o Validation | 78.81 | 6.51 | 72.36 | 5.88 |
| TabAgent | **84.29** | **7.35** | **78.38** | **6.43** |

On the E2E and Rotowire datasets, user queries typically contain general or exploratory extraction instructions. As shown in Figure 3, for E2E and the Team table in Rotowire, TabAgent achieves the best performance when k=7, while for the Player table in Rotowire, the optimal performance is observed at k=5. The initial performance gain stems from the increased contextual support provided by a larger number of memory samples. Knowledge retrieved from the memory repository enables the agents to generate accurate table schema and enhances their capacity for effective self-refinement. However, in the Player table of the Rotowire dataset, performance begins to degrade beyond a certain point, as redundant or noisy information may mislead the agents and negatively impact extraction accuracy. This indicates that more knowledge does not always lead to better outcomes; beyond a threshold, it can become a source of interference.

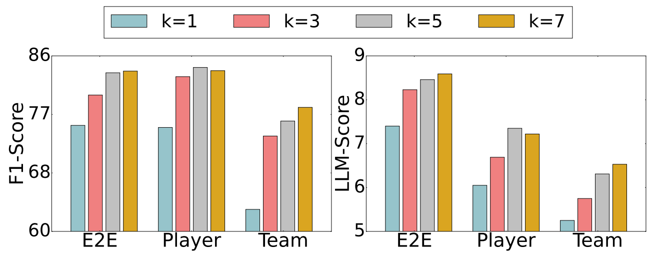


**Figure 3: Effect of top-k values on E2E and Rotowire datasets.**

## 4 RELATED WORK

### 4.1 Structured Information Extraction

Structured information extraction aims to identify and organize structured data from unstructured or semi-structured text. Early approaches relied on rule-based systems to extract entities and attributes by detecting structural linguistic patterns [13, 20, 24, 30]. To improve generalization over complex sentence structures, learning-based methods emerged, leveraging the capabilities of sequence-based and generative models. For example, CharNER [12] applied BiLSTM to named entity recognition, OpenIE6 [11] adopted a BERT-based grid labeling approach for relational triple extraction, and AdaTag [33] introduced adaptive decoder parameterization for multi-attribute extraction. However, these methods often struggle with capturing implicit relationships.

With the advancement of pre-trained models, information extraction has made measurable progress. In particular, when dealing with complex texts with long-term dependencies, pre-trained language models can effectively capture deep semantic information. Text-To-Table [31] fine-tunes the BART [14] model for the table generation task. UIE [17] proposes a unified structure generation framework with a pre-trained large-scale text-to-structure generation model. LasUIE [6] adopts a three-stage LM training procedure to enhance structure-aware generation ability. ODIE [10] is a structured information extractor guided by human instructions, built on LLaMA [28] architecture and fine-tuned via LoRA [9] technique. Although current methods [35, 36] are designed to select valuable data and enhance training efficiency and quality, these training-based models often exhibit limited adaptability, as their performance is constrained by training distributions and predefined schemas. This makes them less flexible when faced with diverse attributes and varying user requirements.

The advent of large language models (LLMs) has introduced new possibilities for information extraction. Empirical results from prior studies [27] demonstrate the capabilities of LLMs in end-to-end table extraction. Systems such as Lotus [23] and Palimpzest [16] exemplify LLM-based frameworks for querying unstructured data, supporting attribute extraction through declarative language. Beyond serving as extractors, LLMs can also play an auxiliary role in extraction tasks. For instance, Revilio [26] employs LLMs to construct table sketches as guiding representations for downstream extraction. Evaporate [3] leverages synthesis code generated by LLMs to perform information extraction from heterogeneous documents. However, their performance is sensitive to factors such as prompt design. These issues can be improved through collaborative mechanisms based on multi-agent systems.

### 4.2 Multi-Agent LLM System For Information Extraction

The integration of multi-agent systems with LLMs has emerged as a promising paradigm for addressing complex tasks. A multi-agent system operates through the collaboration of multiple autonomous agents, each with distinct roles and capabilities. LLM agents demonstrate strong capabilities in diverse scenarios, such as multi-robot system [19], game simulation [32], scientific research [4] and software development [8].

Recently, several studies have introduced multi-agent LLM systems for information extraction and analysis tasks. KARMA [18] decomposes the knowledge graph enrichment pipeline from unstructured text into multiple stages, each managed by specialized agents assigned to distinct roles. Additionally, multi-agent systems can integrate with retrieval-augmented generation frameworks to enhance information extraction by leveraging collaborative reasoning and dynamic knowledge retrieval. MAIN-RAG [5] employs multiple agents to score and filter retrieved documents from knowledge graph before generating responses to user queries. AgentRE [25] proposes an agent-based relation extraction framework, where agents interact with retrieval, memory, and extraction modules to improve relation extraction tasks. MDocAgent [7] performs multimodal extraction and reasoning on textual and visual inputs through the collaboration of several specialized agents. However, these methods face challenges in table extraction tasks due to a lack of explicit mechanisms for schema-aware reasoning, fine-grained cell-level interactions, and strict format verification, which are critical for achieving high precision in structured data processing.

## 5 CONCLUSION

We introduce TabAgent, an LLM-based multi-agent framework designed for structured table extraction from unstructured documents. TabAgent integrates schema generation, iterative self-checking, and structural validation mechanisms via four specialized agents supported by a shared memory repository. Extensive experiments on diverse datasets demonstrate its superior performance compared to existing table extraction methods. Our findings highlight the potential of multi-agent systems in complex structured information extraction tasks. Future work may adapt the framework to various unstructured document formats, including PDFs and image files.

## REFERENCES

[1] [n.d.]. *OpenAI GPT-4o.* https://openai.com/index/hello-gpt-4o/

[2] [n.d.]. *Qwen3.* https://qwenlm.github.io/blog/qwen3/

[3] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433* (2023).

[4] Daniil A Boiko, Robert MacKnight, and Gabe Gomes. 2023. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint arXiv:2304.05332* (2023).

[5] Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, et al. 2024. MAIN-RAG: Multi-Agent Filtering Retrieval-Augmented Generation. *arXiv preprint arXiv:2501.00332* (2024).

[6] Hao Fei, Shengqiong Wu, Jingye Li, Bobo Li, Fei Li, Libo Qin, Meishan Zhang, Min Zhang, and Tat-Seng Chua. 2022. Lasuie: Unifying information extraction with latent adaptive structure-aware generative language model. *Advances in Neural Information Processing Systems* 35 (2022), 15460–15475.

[7] Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. 2025. Mdocagent: A multi-modal multi-agent framework for document understanding. *arXiv preprint arXiv:2503.13964* (2025).

[8] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352* (2023).

[9] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* (2022).

[10] Yizhu Jiao, Ming Zhong, Sha Li, Ruining Zhao, Siru Ouyang, Heng Ji, and Jiawei Han. 2023. Instruct and extract: Instruction tuning for on-demand information extraction. *arXiv preprint arXiv:2310.16040* (2023).

[11] Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Soumen Chakrabarti, et al. 2020. Openie6: Iterative grid labeling and coordination analysis for open information extraction. *arXiv preprint arXiv:2010.03147* (2020).

[12] Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers.* 911–921.

[13] Taesung Lee, Zhongyuan Wang, Haixun Wang, and Seung-won Hwang. 2013. Attribute extraction and scoring: A probabilistic approach. In *2013 IEEE 29th International Conference on Data Engineering (ICDE).* 194–205. https://doi.org/10.1109/ICDE.2013.6544825

[14] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[15] Qi Li and Heng Ji. 2014. Incremental Joint Extraction of Entity Mentions and Relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Baltimore, Maryland, 402–412. https://doi.org/10.3115/v1/P14-1038

[16] 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 preprint arXiv:2405.14696* (2024).

[17] Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. *arXiv preprint arXiv:2203.12277* (2022).

[18] Yuxing Lu and Jinzhuo Wang. 2025. KARMA: Leveraging Multi-Agent LLMs for Automated Knowledge Graph Enrichment. *arXiv preprint arXiv:2502.06472* (2025).

[19] Zhao Mandi, Shreeya Jain, and Shuran Song. 2024. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 286–299.

[20] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open Language Learning for Information Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.* Association for Computational Linguistics, Jeju Island, Korea, 523–534. https://aclanthology.org/D12-1048/

[21] Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Berlin, Germany, 1105–1116. https://doi.org/10.18653/v1/P16-1105

[22] Makoto Miwa and Yutaka Sasaki. 2014. Modeling Joint Entity and Relation Extraction with Table Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Association for Computational Linguistics, Doha, Qatar, 1858–1869. https://doi.org/10.3115/v1/D14-1200

[23] Liana Patel, Siddharth Jha, Carlos Guestrin, and Matei Zaharia. 2024. Lotus: Enabling semantic queries with llms over tables of unstructured and structured data. *arXiv preprint arXiv:2407.11418* (2024).

[24] Swarnadeep Saha and Mausam. 2018. Open Information Extraction from Conjunctive Sentences. In *Proceedings of the 27th International Conference on Computational Linguistics.* Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2288–2299. https://aclanthology.org/C18-1194/

[25] Yuchen Shi, Guochao Jiang, Tian Qiu, and Deqing Yang. 2024. AgentRE: An Agent-Based Framework for Navigating Complex Information Landscapes in Relation Extraction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management.* 2045–2055.

[26] Mukul Singh, Gust Verbruggen, Vu Le, and Sumit Gulwani. 2024. Tabularis Revilio: Converting Text to Tables. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management.* 4056–4060.

[27] Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. 2023. Struc-bench: Are large language models really good at generating complex structured data? *arXiv preprint arXiv:2309.08963* (2023).

[28] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[30] Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal Decompositional Semantics on Universal Dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, Austin, Texas, 1713–1723. https://doi.org/10.18653/v1/D16-1177

[31] Xueqing Wu, Jiacheng Zhang, and Hang Li. 2021. Text-to-table: A new way of information extraction. *arXiv preprint arXiv:2109.02707* (2021).

[32] Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023. Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658* (2023).

[33] Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. 2021. Adatag: Multi-attribute value extraction from product profiles with adaptive decoding. *arXiv preprint arXiv:2106.02318* (2021).

[34] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR).*

[35] Chi Zhang, Huaping Zhong, Hongtao Li, Chengliang Chai, Jiawei Hong, Yuhao Deng, Jiacheng Wang, Tian Tan, Yizhou Yan, Jiantao Qiu, Ye Yuan, Guoren Wang, Conghui He, and Lei Cao. 2025. Not All Documents Are What You Need for Extracting Instruction Tuning Data. arXiv:2505.12250 [cs.CL] https://arxiv.org/abs/2505.12250

[36] Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Jiantao Qiu, Lei Cao, Ju Fan, Ye Yuan, Guoren Wang, and Conghui He. 2024. Harnessing Diversity for Important Data Selection in Pretraining Large Language Models. arXiv:2409.16986 [cs.AI] https://arxiv.org/abs/2409.16986

[37] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).