AID-Agent: An LLM-Agent for Advanced Extraction and Integration of Documents

Bin Li, Jannis Conen, Felix Aller
cbs Corporate Business Solutions, Heidelberg Germany
{bin.li, jannis.conen, felix.aller}@cbs-consulting.de

Abstract

Extracting structured information from complex unstructured documents is an essential but challenging task in today's industrial applications. Complex document content, e.g., irregular table layout, and cross-referencing, can lead to unexpected failures in classical extractors based on Optical Character Recognition (OCR) or Large Language Models (LLMs). In this paper, we propose the AID-agent framework that synergistically integrates OCR with LLMs to enhance text processing capabilities. Specifically, the AID-agent maintains a customizable toolset, which not only provides external processing tools for complex documents but also enables customization for domain and task-specific tool requirements. In the empirical validation on a real-world use case, the proposed AID-agent demonstrates superior performance compared to conventional OCR and LLM-based approaches.

1 Introduction

Automatic extraction and integration of document information is an essential task for digitalization. Unstructured data in PDF documents are traditionally manually extracted and integrated into a structured schema. The development of advanced OCR techniques leverages the automation of machine extractions in many scenarios; nevertheless, the information recognized by OCR engines remains at a synthetic and symbolic level, which leads to massive failures in complex documents, where the target information is not explicitly extractable. Even some commercial tools (e.g., Azure Document Intelligence [1], Google Document AI [2]) achieve superior OCR performance on general-purpose tasks, they may still fail when applied to domain-specific and complex documents.

Recently, multi-modal LLMs support directly prompting PDF documents as images and extracting information in a question-answering fashion. However, two major challenges hinder the usage of such models at scale. Firstly, a complex document usually contains unstructured information (e.g., irregular table structure) or cross-references within the document, meaning that feeding the OCR-parsed text into an LLM often leads to suboptimal results. Although multimodal LLMs can incorporate layout information and prompt long context, at the current stage, they still result in a significant cost overhead than classical approaches. It is especially inefficient for long documents, where the relevant information only lies on a few pages. Secondly, despite recent advances in the long-context feature of LLMs, holistically resolving target information from multiple spots in complex documents is still a challenging task. Furthermore, in some domain-specific tasks, extracting professional entities is also beyond the capability of general-purpose LLMs, e.g., pictograms, chemical compositions, and technical abbreviations.

To this end, we propose using LLM-based agents to handle the extraction of complex documents, where external tools with centralized scheduling efficiently optimize the extraction process. Agent AI systems enable AI models to solve complex tasks independently. Specifically empowered by recent LLMs, LLM-agents have shown superior capability in several application domains (Durante et al., 2024), e.g., gaming, robotics, and NLP. However, in the document information extraction domain, the research is still underexplored. A desired LLM-agent is supposed to extract document information and integrate it into the target schema in an end-to-end fashion. Task decomposition, reasoning, and scheduling are key abilities in this scenario. General and task-specific tools can accelerate the effective extraction procedure. Finally, an internal validator should ensure consistency and validity

[1] https://azure.microsoft.com/en-us/products/ai-services/ai-document-intelligence
[2] https://cloud.google.com/document-ai

Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025), pages 80–88

based on available knowledge, in either a zero-shot or a few-shot setting.

In this paper, we propose the AID-agent (Advanced Integration of Documents-Agent), an LLM-agent specialized in complex document information extraction. Our contribution can be summarized as follows:

- we present the architecture of AID-agent for complex document extraction and integration,
- we introduce multiple general and task-specific external tools that facilitate the agent processing capability,
- we explore potential improvement possibilities in training the AID-agent scheduling strategy and enabling full-automatic agent decision-making.

2 Related work

2.1 Agentic approaches

A few recent works focus on building static workflows or dynamic agent systems using LLMs for document extraction tasks. Wiest et al. (Wiest et al., 2024) propose LLM-AIx, an LLM-based pipeline for unstructured medical extraction, where LLM extraction is processed with static evaluation modules. Instead of extracting information to a predefined schema, Musumeci et al. (Musumeci et al., 2024) utilize a multi-agent system to generate a desired document, in which the agents can communicate with each other on the current requirement. Luo et al. (Luo et al., 2024) decompose the conventional extraction workflow into three agents, schema agent, extraction agent, and reflection agent. Jiao et al. (Jiao et al., 2024) propose an agent framework with multiple data science tools for the extraction of structured tables. Despite existing work on agent-based document extraction, dealing with complex documents remains to be explored.

2.2 Non-agentic approaches

Several document extraction works have already adopted a single LLM with certain prompting in their solution frameworks. However, without the agentic style reasoning and executing external tools, the quality of such approaches is limited to the LLM's capability and prompting. Perot (Perot et al., 2024) et al. proposed LMDX, which utilizes the OCR-parsed text and their spatial position for the extraction and localization of information. They apply a chunking step to fit long context into the LLM input, however, the model is not optimized toward complex information like cross-references. Fornasiere et al. employ LLM for medical document extraction. Specifically, they proposed a timeline-fashion extraction, where the LLM can extract a sequence of documents corresponding to a patient's clinical history. Furthermore, there are several works on LLM-based structured document extraction. For instance, entity linking from scientific tables (Oshima et al., 2024) and Text-to-SQL (Tai et al., 2023). In our problem setting, we aim to make use of LLMs as the core engine of processing both structured and unstructured documents and embed them in an agent system for efficient scheduling and reasoning.

3 AID-agent

In this section, we introduce the proposed AID-agent, i.e., an LLM-agent for advanced extraction and integration of documents. The architecture of AID-agent is visualized in Figure 1.

The AID-agent is designed to process complex document extraction tasks end-to-end. It accepts PDF documents and the target extraction schema in JSON format as inputs and outputs the filled JSON schema after extraction. Internally, AID-agent consists of three major components, an executor, a tool pool, and a validator.

The executor is implemented using an LLM, which is responsible for task decomposition, scheduling, and tool calling. This can be implemented, for example, using the ReAct (Yao et al., 2022) framework for efficient reasoning and action.

The tool pool comprises a set of external tools for the executor. Depending on the document property, the executor automatically selects tools to solve the extraction task. In the current stage, the automatic tool selection is based on the initial prompt, tool description as well as validator feedback. Ultimately, we aim to train the executor using reinforcement learning for an optimal tool selection, refer to Section 5 for more discussion. Aiming at complex documents that lead to failures of naive LLM-based approaches, we introduce three general tools in our tool pool. The table resolver organizes all structured tables in the document, which enables retrieving tabular information in an SQL-fashion. The reference resolver targets cross-referenced information in the document and integrates multi-

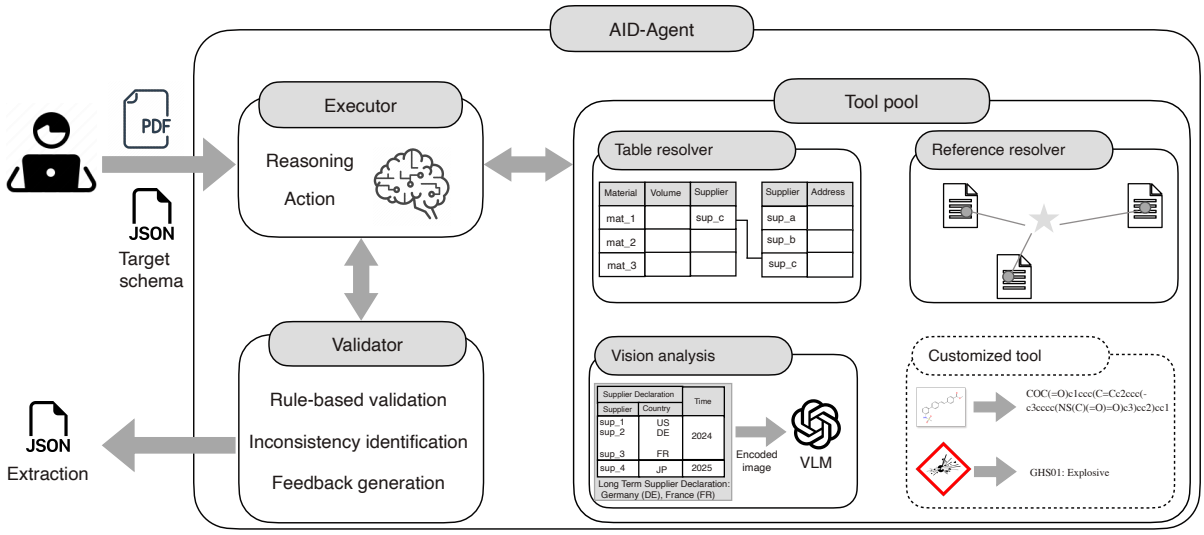

Figure 1: AID-agent architecture

ple information sources in the document to generate the final answer. It is implemented using keyword matching and semantic embedding comparison. Both table resolver and reference resolver return the retrieved relevant information back to the executor for the LLM-based extraction. For certain documents with irregular and unstructured content, e.g., irregular tables with attached text descriptions, the vision analysis tool provides a powerful option to identify the bounding box of the target document area and use the vision language model (VLM) for the extraction. Since only irregular, complex parts of the document are extracted using VLM, we limit the expense as much as possible. Finally, for domain- and task-specific demands, we also leave a standard interface in the tool pool for customized tools, where specific professional tools can be implemented to leverage the LLM-extraction, e.g., chemical composition translator, and pictogram checklist. In the few-shot setting, a few extraction examples or domain expert guidance can also be included as a customized tool, which provides additional knowledge and regulation for the extraction.

Lastly, the validator interacts with the executor and provides an assessment of the current extraction quality, which will guide the tool selection action in the next iteration. A rule-based function validates compatibility of extraction formatting w.r.t. the target schema. Furthermore, the consistency among similar extraction fields is ensured by using regular expressions. Given the target schema and the current extraction, we also use LLM-as-a-judge to generate feedback to indicate possible improvements for the executor. Once all fields are extracted (missing fields marked as None), or the agent reaches the predefined maximum iterations, the extraction procedure terminates.

4 Case study

We present a case study of applying the AID-agent on technical report documents from our industrial partner. Specifically, the goal is to extract metadata of the supplier information and chemical compositions of materials from each document. A toy example of the procedue is demonstrated in Appendix A. We have collected 44 manually labeled PDF documents for evaluation. The major challenges are that different suppliers organize the document in different layouts, and the information is presented in irregular table structures, causing classical OCR and table detection techniques to fail. Furthermore, the target chemical composition volume number is usually surrounded by confusing items, like the common volume range, which makes it not trivial to extract without proper row and column headers in the tables.

We include a domain expert knowledge tool to provide extra knowledge for the agent. The tool loads a file that contains field types and allowed data formats. Additionally, we introduce a table extraction tool, which resolves and reconstructs the table structure from the OCR-parsed text. Specifically, it supports the LLM to understand table structure by analyzing the row and column alignment and incorporating the table cell coordinates.

The PDF documents passed to the agent are

Model	Accuracy	Precision	Recall	F1-Score
Baseline	0.767 ± 0.242	0.807 ± 0.248	0.897 ± 0.209	0.839 ± 0.224
w/o table extractor	0.794 ± 0.185	0.842 ± 0.189	0.911 ± 0.152	0.869 ± 0.165
w/o VLM	0.800 ± 0.198	0.868 ± 0.169	0.889 ± 0.193	0.870 ± 0.174
w/o validator	0.824 ± 0.204	0.875 ± 0.207	0.893 ± 0.206	0.882 ± 0.203
AID-agent (gpt-4o-mini)	0.673 ± 0.211	0.786 ± 0.193	0.788 ± 0.193	0.782 ± 0.187
AID-agent (gpt-4o)	**0.867 ± 0.093**	**0.916 ± 0.082**	**0.941 ± 0.058**	**0.926 ± 0.059**

Table 1: Performance and ablation study

firstly parsed into text and layout information using the Azure Document Intelligence API. Specifically, we introduce line-entity, which organizes each OCR-identified content block as an entity line-wise. Each line-entity is represented by $\{Content \,|\, PageNum \,|\, Coordinate \,|\, ContentType\}$.The content type includes text, title, image, and table. Specifically for tables, we append $\{TableNum \,|\, RowNum \,|\, ColumnNum\}$. With this, we include all necessary information from the OCR result, while consuming significantly fewer tokens in the prompt compared to classical markdown or JSON representations.

As evaluation metrics, we employ accuracy, precision, recall, and F1-score, while true positive is the number of correct extracted fields, false positive is the number of mistaken but non-empty fields and false negative is the number of mistaken empty fields. We evaluate the metrics per document and report the overall average and standard deviation. Furthermore, in order to assess the effectiveness of tools employed by the agent, we also conduct multiple ablation studies that exclude one tool at a time, as well as replace the standard GPT-4o model with GPT-4o-mini model as the base model. And we include "OCR+LLM" as a baseline solution where no additional tool is available. The evaluation results are summarized in Table 1.

Generally, the AID-agent with complete tool access performs the best. With a 0.926 F1-score, the agent is able to extract the majority of desired information. The reasons for a few representative mistakes are poor OCR recognition, misalignment of complex tables, and integration of multiple extraction versions. Since we did not include any treatment to rotated content, the performance on one $90°$-rotated document is significantly lower than others, even feeding the cropped image to the VLM did not improve the result.

From the various ablation studies, we determined the necessity of the toolset we designed. Leaving out any of the tools will result in decreased performance. The most significant performance degradation is caused by replacing the base LLM from GPT-4o to GPT-4o-mini. This result indicates that the necessary information is included either in the OCR-parsed text or the cropped images, and a sub-optimal LLM or VLM can misuse it in the agent inference process.

5 Discussion

In this paper, we have introduced the AID-agent for complex document extraction and integration. Multiple tools and the validator support efficient reasoning and action of the executor during processing. The customizable tool pool also enables extensive extension flexibility for use case-specific challenges. Compared to classical LLM-based document extraction, which requires enormous effort in prompting engineering, the prompts used by the AID-agent executor and tools are reusable, while task-specific information can be included as a tool.

The major improvement we are trying to bring to the AID-agent is the optimization of tool scheduling. At the current stage, the executor applies reasoning and action steps based on the extraction task in the prompts and tool descriptions. The LLM of the agent executor makes the scheduling using all textual information within the agent. In the future, we aim to train the executor using a few-shot reinforcement learning approach. Specifically, with labeled samples, the executor is supposed to optimize the order of tool calls to minimize the time and API-call costs. For instance, directly reading selectable PDFs is more time-efficient than OCR, and text-based LLM extraction is more cost-efficient than image-based VLM extraction. These can be regularized as a reward function.

At the current stage, we have conducted a case

study using a limited set of proprietary PDF documents from a single domain. As a next step, we plan to evaluate the AID-agent on various public benchmark datasets, such as BIGDOCS (Rodriguez et al., 2025). Our primary focus is to facilitate the automation of industrial document processing tasks, including invoices, purchase orders, and delivery notes. Furthermore, we aim to conduct comparative studies with recent document extraction models. Specifically, we intend to compare AID-agent with pipeline-based extraction tools (e.g., MinerU (Wang et al., 2024), Docling (Livathinos et al., 2025)) that perform step-by-step processing—from layout analysis and OCR to text integration—as well as with end-to-end VLMs (e.g., GOT-OCR (Wei et al., 2024), SmolDocling (Nassar et al., 2025)) that directly produce structured outputs from input documents. Finally, we also consider leveraging state-of-the-art LLMs, such as Qwen (Yang et al., 2025) and LLaMA (Touvron et al., 2023), as potential backbones for AID-agent.

References

Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, Katsushi Ikeuchi, Hoi Vo, Li Fei-Fei, and Jianfeng Gao. 2024. Agent AI: Surveying the Horizons of Multimodal Interaction. *arXiv preprint*. ArXiv:2401.03568 [cs].

Yizhu Jiao, Sha Li, Sizhe Zhou, Heng Ji, and Jiawei Han. 2024. Text2DB: Integration-Aware Information Extraction with Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 185–205, Bangkok, Thailand. Association for Computational Linguistics.

Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Val'ery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2025. Docling: An efficient open-source toolkit for ai-driven document conversion. *ArXiv*, abs/2501.17887.

Yujie Luo, Xiangyuan Ru, Kangwei Liu, Lin Yuan, Mengshu Sun, Ningyu Zhang, Lei Liang, Zhiqiang Zhang, Jun Zhou, Lanning Wei, Da Zheng, Haofen Wang, and Huajun Chen. 2024. OneKE: A Dockerized Schema-Guided LLM Agent-based Knowledge Extraction System. *arXiv preprint*. ArXiv:2412.20005 [cs] version: 1.

Emanuele Musumeci, Michele Brienza, Vincenzo Suriani, Daniele Nardi, and Domenico Daniele Bloisi. 2024. Llm based multi-agent generation of semi-structured documents from semantic templates in the public administration domain. In *Interacción*.

Ahmed Nassar, Andrés Marafioti, Matteo Omenetti, Maksym Lysak, Nikolaos Livathinos, Christoph Auer, Lucas Morin, Rafael Teixeira de Lima, Yusik Kim, A. Said Gurbuz, Michele Dolfi, Miquel Farr'e, and Peter W. J. Staar. 2025. Smoldocling: An ultra-compact vision-language model for end-to-end multi-modal document conversion. *ArXiv*, abs/2503.11576.

Yuji Oshima, Hiroyuki Shindo, Hiroki Teranishi, Hiroki Ouchi, and Taro Watanabe. 2024. Synthetic Context with LLM for Entity Linking from Scientific Tables. In *Proceedings of the Fourth Workshop on Scholarly Document Processing (SDP 2024)*, pages 202–214, Bangkok, Thailand. Association for Computational Linguistics.

Vincent Perot, Kai Kang, Florian Luisier, Guolong Su, Xiaoyu Sun, Ramya Sree Boppana, Zilong Wang, Zifeng Wang, Jiaqi Mu, Hao Zhang, Chen-Yu Lee, and Nan Hua. 2024. LMDX: Language Model-based Document Information Extraction and Localization. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 15140–15168, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Juan Rodriguez, Xiangru Jian, Siba Smarak Panigrahi, Tianyu Zhang, Aarash Feizi, Abhay Puri, Akshay Kalkunte, François Savard, Ahmed Masry, Shravan Nayak, Rabiul Awal, Mahsa Massoud, Amirhossein Abaskohi, Zichao Li, Suyuchen Wang, Pierre-André Noël, Mats Leon Richter, Saverio Vadacchino, Shubham Agarwal, and 24 others. 2025. Bigdocs: An open dataset for training multimodal models on document and code tasks. *Preprint*, arXiv:2412.04626.

Chang-You Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain-of-thought style prompting for text-to-sql. *ArXiv*, abs/2305.14215.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Bin Wang, Chaochao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. 2024. Mineru: An open-source solution for precise document content extraction. *ArXiv*, abs/2409.18839.

Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jian-Yuan Sun, Yuang Peng, Chunrui Han, and Xiangyu Zhang. 2024. General ocr theory: Towards ocr-2.0 via a unified end-to-end model. *ArXiv*, abs/2409.01704.

Isabella C. Wiest, Fabian Wolf, Marie-Elisabeth Leßmann, M. van Treeck, Dyke Ferber, Jiefu Zhu, Heiko Boehme, Keno Kyrill Bressem, Hannes Ulrich, M. P. Ebert, and J. N. Kather. 2024. Llm-aix: An open source pipeline for information extraction from unstructured medical text based on privacy preserving large language models. *medRxiv*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629.

A Appendix

A conversational example of AID-agent

User: Extract following schema from the document ./demo.pdf
SCHEMA:

```
{
    "materialNumber": null,
    "batchNumber": null,
    "chemicalComposition": {
        "C": null,
        "SI": null,
        "P": null
    }
}
```

Agent Thinking: I need to extract the required information from the specified PDF document and utilize the external knowledge to enhance the extraction process. I will start by loading the external knowledge to understand the expected fields better.

Agent Action: load_external_knowledge

Observation:
materialNumber: `^[A-Za-z0-9\-/]+$`
batchNumber: `^\d+$`
chemicalComposition: `{^\d+(\.\d+)?$}`

Agent Thinking: I now need to parse the PDF document to extract the relevant information according to the schema provided.

Agent Action: parse_pdf

Observation:

ID	Content	Page	Coordinates	OCR conf.	Type
0	Material number	1	1017.223,43.394	0.996	text
...					
50	C	1	1186.075,466.136	0.935	table 0 0 0
51	SI	1	1595.405,465.859	0.997	table 0 0 1
52	P	1	2004.103,465.981	0.843	table 0 0 2
53	0.2	1	1185.976,503.339	0.892	table 0 1 0
54	$> 0, <0.5$	1	1595.342,503.972	0.601	table 0 1 1
...					

Agent Thinking: Based on the provided text, the SCHEMA can be filled as follows:

```
{
    "materialNumber": 99-001002/003/A,
    "batchNumber": null,
    "chemicalComposition": {
        "C": 0.2,
        "SI": >0, <0.5,
        "P": 0.1
    }
}
```

Agent Action: validator

Agent Thinking: The validator has flagged the "chemicalComposition.SI" field due to format conflict with external knowledge. I'll re-run the extraction using the "**vision_analysis**" on the cropped areas where the chemical composition is likely to be found, as this may help in retrieving more accurate numeric values.

Agent Action: vision_analysis

Observation Based on the provided images, the SCHEMA can be filled as follows:

```
{
    "materialNumber": 99-001002/003/A,
    "batchNumber": null,
    "chemicalComposition": {
        "C": 0.2,
        "SI": 0.25,
        "P": 0.1
    }
}
```

Agent Thinking: I now have refined values for the extracted fields. The new extraction results are more consistent, particularly for the "chemicalComposition" field, which now contains only numeric values as expected.

Final Answer:


```
{
    "materialNumber": 99-001002/003/A,
    "batchNumber": null,
    "chemicalComposition": {
        "C": 0.2,
        "SI": 0.25,
        "P": 0.1
    }
}
```