DocBench: A Benchmark for Evaluating LLM-based Document Reading Systems

Anonymous ARR submission

Abstract

Recently, there has been a growing interest among large language model (LLM) developers in LLM-based document reading systems, which enable users to upload their own documents and pose related questions, addressing challenges like file parsing, metadata extraction, multi-modal information understanding, and long-context reading. However, no current benchmark exists to evaluate their performance in such scenarios, where a raw file and questions are provided as input, and a corresponding response is expected as output. In this paper, we introduce DocBench, a new benchmark designed to assess LLM-based document reading systems. It includes 229 real documents and 1,102 questions across five domains and four major question types, created through human annotators and synthetic question generation. Our findings highlight significant gaps between existing LLM-based document reading systems and human performance, emphasizing the challenges in developing proficient systems. DocBench aims to standardize the evaluation of these systems in diverse real-world scenarios, guiding future advancements in this field.

1 Introduction

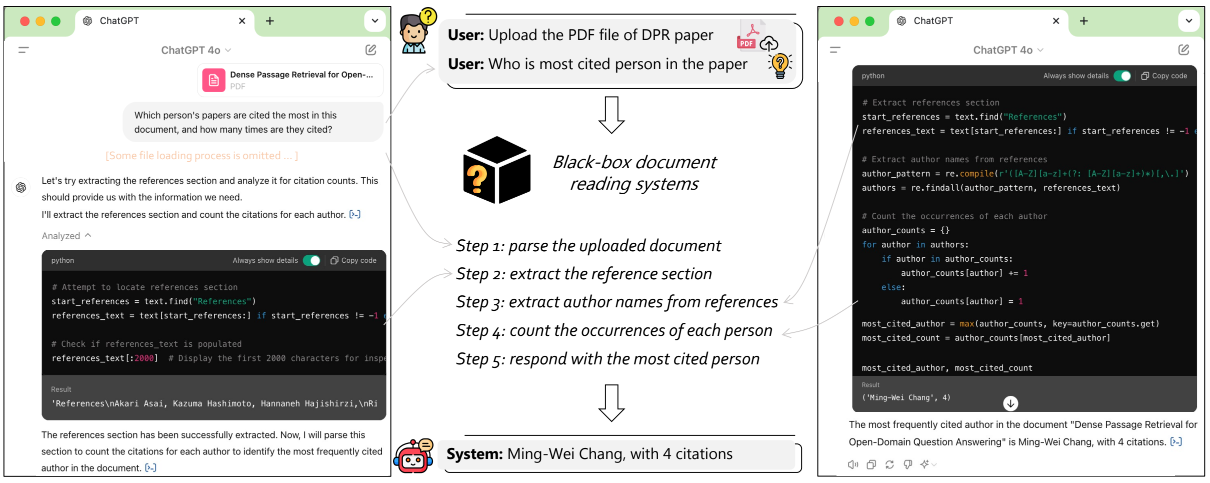
The emergence of large language models (LLMs) has marked a significant milestone in the field of natural language processing, revolutionizing the way we approach a variety of tasks (Zhao et al., 2023; Chang et al., 2024; Wang et al., 2024a; Achiam et al., 2023; Anthropic, 2024; Touvron et al., 2023; Team et al., 2023). Existing LLMs such as GPT-4 (Achiam et al., 2023), Llama-3 (Touvron et al., 2023), and Claude-3 (Anthropic, 2024) have shown exceptional abilities in following human instructions to perform tasks such as answering questions, translating languages and summarizing texts. These tasks are typically characterized by straightforward input-output interactions, where the models generate responses solely based on the provided text. However, many real-world applications require more complex interactions involving user-provided documents. For instance, financial analysts might need to query comprehensive financial reports to inform their investment decisions (Yang et al., 2023; Wu et al., 2023). Legal professionals often search through extensive legal documents to find relevant case law (Lai et al., 2023; Cui et al., 2023). Similarly, scientific researchers frequently sift through academic papers to identify related works and extract key findings (Dasigi et al., 2021; Birhane et al., 2023).

When users pose queries based on their provided documents, the situation becomes more intricate and challenging (Lee et al., 2024). Unlike standalone LLMs that are primarily trained to process and respond to textual inputs (or images in the case of Vision LLMs), handling user-provided documents necessitates a more sophisticated approach that stretches beyond the capabilities of a single LLM. In order to provide accurate responses, an LLM-based document reading system should not only comprehend natural language queries, but also excel in a range of processing skills, including parsing and interpreting user documents and layouts, navigating complex formatting structures, extracting relevant metadata, and managing long textual contexts along with any embedded images. Mastery of these diverse skills is essential for generating precise and contextually relevant responses.

At the same time, recent advancements in proprietary LLM developers such as OpenAI and Anthropic have provoked the release of several LLM-based document reading systems. Figure 1 illustrates an example of OpenAI's GPT-4-based document reading system. Despite widespread claims of effectiveness and efficiency in various online public blogs[1][2], **the absence of a standardized**

[1]Blog: Claude can now use tools `https://www.anthropic.com/news/tool-use-ga`

[2]Blog: Talk with documents using Lla-



Which person's papers are cited the most in this document, and how many times are they cited?

[Some file loading process is omitted ...]

Let's try extracting the references section and analyze it for citation counts. This should provide us with the information we need.
I'll extract the references section and count the citations for each author.

```python
# Attempt to locate references section
start_references = text.find("References")
references_text = text[start_references:] if start_references != -1

# Check if references_text is populated
references_text[:2000]  # Display the first 2000 characters for inspe
```

```
Result
'References\nAkari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi,\nRi
```

The references section has been successfully extracted. Now, I will parse this section to count the citations for each author to identify the most frequently cited author in the document.

User: Upload the PDF file of DPR paper
User: Who is most cited person in the paper

Black-box document reading systems

Step 1: parse the uploaded document
Step 2: extract the reference section
Step 3: extract author names from references
Step 4: count the occurrences of each person
Step 5: respond with the most cited person

System: Ming-Wei Chang, with 4 citations

```python
# Extract references section
start_references = text.find("References")
references_text = text[start_references:] if start_references != -1

# Extract author names from references
author_pattern = re.compile(r'([A-Z][a-z]+(?: [A-Z][a-z]+)*)[,\.]')
authors = re.findall(author_pattern, references_text)

# Count the occurrences of each author
author_counts = {}
for author in authors:
    if author in author_counts:
        author_counts[author] += 1
    else:
        author_counts[author] = 1

most_cited_author = max(author_counts, key=author_counts.get)
most_cited_count = author_counts[most_cited_author]

most_cited_author, most_cited_count
```

```
Result
('Ming-Wei Chang', 4)
```

The most frequently cited author in the document "Dense Passage Retrieval for Open-Domain Question Answering" is Ming-Wei Chang, with 4 citations.

Figure 1: An example of OpenAI's GPT-4 based document reading system. Unlike standalone LLMs, recent proprietary LLM-based document reading systems employ a carefully designed approach (e.g., file parsing, code execution) to answer user questions related to document contents.

benchmark makes it difficult to objectively evaluate and compare the document reading performance across these systems, thereby leaving a critical gap in fairly assessing these capabilities in a fine-grained manner.

To fill this gap, our paper introduces DocBench, a novel benchmark specifically designed to evaluate LLM-based document reading systems. DocBench is developed to mirror real-world scenarios where each input consists of a document paired with one or multiple associated questions, and each question is annotated with a golden answer. Our benchmark undergoes a meticulous development process, incorporating human annotation and synthetic question generation. To the end, DocBench features 229 real-world documents and 1,102 questions spanning 5 diverse domains: *Academia, Finance, Government, Laws, and News*. Besides, the benchmark involves 4 question categories, including *text-only, multi-modal (i.e., tables and figures), meta-data, and unanswerable*, ensuring comprehensive coverage of various document reading capabilities.

Based upon DocBench, we evaluate several proprietary LLM-based systems that are accessible via web interfaces or APIs. However, these proprietary systems are close-sourced, thus leading to the limited disclosure of their detailed operational strategies. As a result, we additionally assess a straightforward parse-then-read pipeline employing a series of open-source LLMs. Our evaluations reveal noticeable gaps between existing LLM-based document reading systems and human performance, underscoring the challenges of developing proficient systems.

In summary, DocBench serves as the first standardized benchmark to evaluate LLM-based document reading systems within real-world scenarios, where the systems take a document file paired with one or multiple related questions as input and generate textual responses as output. Moreover, our benchmark is carefully designed to encompass 5 diverse domains and 4 distinct question types, ensuring a nuanced and thorough assessment. By facilitating fair comparisons across different systems, DocBench highlights current limitations and paves the way for future advancements.

2 The DocBench

DocBench is a benchmark that takes raw PDF files and accompanying questions as inputs, with the objective of generating corresponding textual answers. In this section, we will introduce the pipeline used to construct the dataset, present detailed statistics, and explain the evaluation method.

2.1 Dataset Construction

Our dataset construction pipeline consists of three phases. First, we crawl documents across various domains from publicly accessible online resources (§2.1.1). Second, we generate corresponding QA pairs with the help of GPT-4 and a team of human annotators (§2.1.2). Finally, we conduct auto filtering followed by a manual review to validate the quality of the generated instances (§2.1.3).

maIndex https://codemaker2016.medium.com/talk-with-documents-using-llamaindex-3952c76bd511

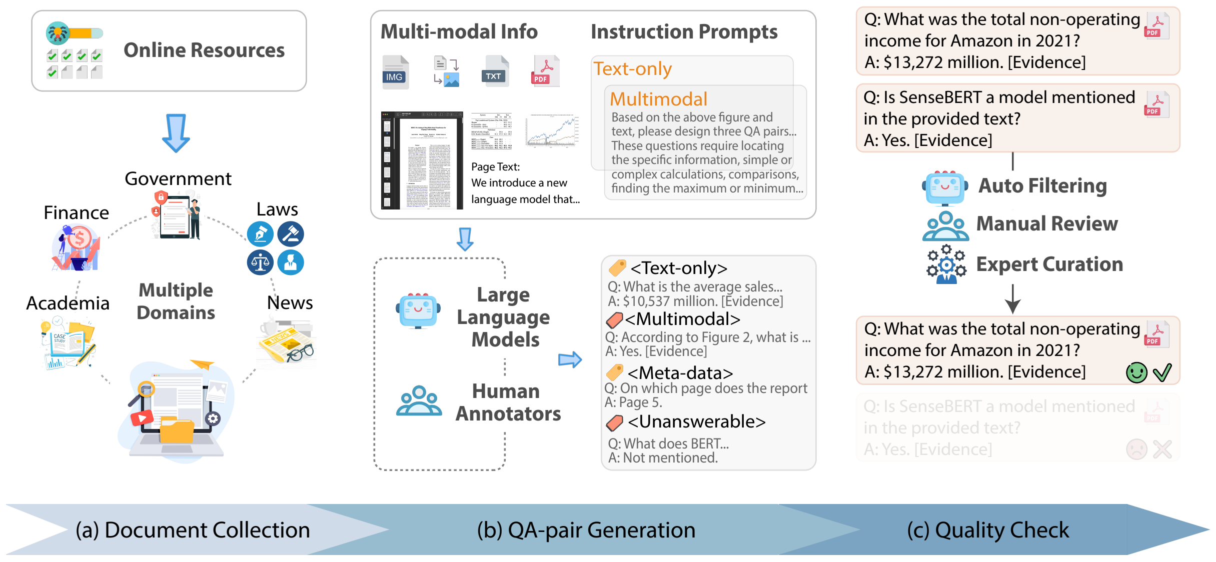

Online Resources
Government
Finance
Laws
Academia
Multiple Domains
News
Multi-modal Info
Instruction Prompts
Text-only
Multimodal
Based on the above figure and text, please design three QA pairs... These questions require locating the specific information, simple or complex calculations, comparisons, finding the maximum or minimum...
Page Text: We introduce a new language model that...
Large Language Models
Human Annotators
<Text-only>
Q: What is the average sales... A: $10,537 million. [Evidence]
<Multimodal>
Q: According to Figure 2, what is ... A: Yes. [Evidence]
<Meta-data>
Q: On which page does the report A: Page 5.
<Unanswerable>
Q: What does BERT... A: Not mentioned.
Q: What was the total non-operating income for Amazon in 2021? A: $13,272 million. [Evidence]
Q: Is SenseBERT a model mentioned in the provided text? A: Yes. [Evidence]
Auto Filtering
Manual Review
Expert Curation
Q: What was the total non-operating income for Amazon in 2021? A: $13,272 million. [Evidence]
Q: Is SenseBERT a model mentioned in the provided text? A: Yes. [Evidence]
(a) Document Collection
(b) QA-pair Generation
(c) Quality Check

Figure 2: Construction pipeline of DOCBENCH. (a) Document Collection: gathering PDF files from five different domains; (b) QA-pair Generation: creating diverse and comprehensive QA pairs through a combination of LLMs and human effort; (c) Quality Check: ensuring data quality through a multi-step process.

2.1.1 Document Collection

To establish a practical and constructive benchmark for document reading, we concentrate on scenarios where it is crucial to read documents. We standardize the documents to PDF format due to its high compatibility and stability. We identify five domains where documents are frequently utilized: *Academia*, *Finance*, *Government*, *Laws*, *News*. For Academia, papers are downloaded from arXiv within the range of top-k citations in the field of natural language processing on Google Scholar. [3] For *Finance*, we crawl the annual reports of companies with top-k global market capitalization up to `2024-02-23` from AnnualReports. [4] For *Government*, we manually download official governmental reports in 2023 from the U.S. Department of State and GovInfo. [5] For *Laws*, files are gathered from an official online collection of publications from the Library of Congress, within the years ranging from 2020 to 2024. [6] For *News*, we collect front-page scanned documents of the New York Times, covering dates from `2022-02-22` to `2024-02-22`. [7] We set $k = 100$ in the initial crawling process for academic and financial documents. After skipping the unobtainable or damaged documents, we eventually obtained 229 PDF files, with 49 for academia, 40 for finance, 44 for government, 46 for laws, and 50 for news. Detailed statistics are shown in Table 1.

[3] https://scholar.google.com/; https://arxiv.org/.
[4] https://companiesmarketcap.com; http://www.annualreports.com.
[5] https://www.state.gov/department-reports/; https://www.govinfo.gov/.
[6] https://www.loc.gov/collections/publications-of-the-law-library-of-congress.
[7] https://static01.nyt.com/images/.

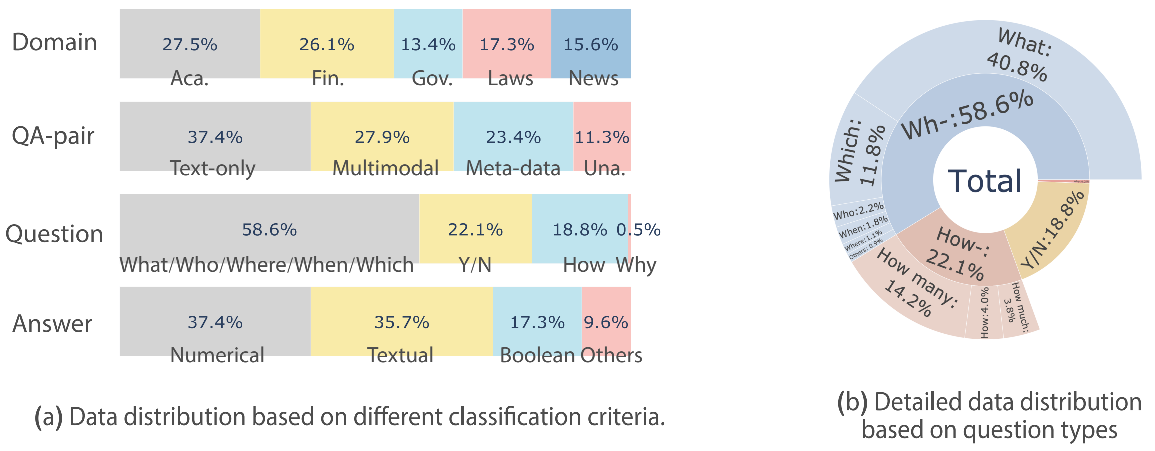
2.1.2 QA-pair Generation

The generation procedure revolves around two aspects: diversity and comprehensiveness. On one hand, as the document itself inherently abounds with multi-dimensional and multi-modal information including texts, tables, figures, and meta-data, we leverage the `fitz` library [8] to parse out the distinct modalities within the PDF files. Afterward, we deliver plain texts to GPT-4 (`gpt-4-0125-preview`) for generating *text-only* QA pairs and resort to GPT-4V (`gpt-4-1106-vision-preview`) for yielding multi-modal ones based on tables, figures, and their related textual descriptions. On the other hand, we further request a set of human annotators to manually elaborate 350 QA pairs based on the given document files. Their primary task is to focus on types that are rarely covered in the previous generation stage but are frequent in daily usage, such as meta-data and unanswerable instances. Details of the annotation process and instruction prompts are attached in Appendix B.

[8] https://pypi.org/project/fitz/

Category	Questions.		Documents.			
	#Num	#Tokens	#Num	#Pages	#Size(KB)	#Tokens
Aca.	303	16.8	49	11	847	11,123
Fin.	288	16.8	40	192	6,594	149,409
Gov.	148	14.1	44	69	2,183	36,105
Laws	191	15.4	46	58	969	32,339
News	172	13.5	50	1	3,095	2,909
Total/Avg.	1,102	15.7	229	66	2,738	46,377

Table 1: Overview statistics of DocBench. All documents are in PDF format. We extract text content and calculate the corresponding *#Tokens* of documents.



(a) Data distribution based on different classification criteria.

(b) Detailed data distribution based on question types

Figure 3: Data distribution of DocBench: (a) proportion(%) of various data groups based on four distinct classification criteria; (b) detailed data analysis based on question types.

2.1.3 Quality Check

We begin by instructing GPT-4 to automatically filter out questions that are excessively lengthy, unnatural, or impractical. We then conduct a manual review following the automatic filtering to ensure both the quality of questions and the accuracy of answers. To further align our data with real-world user scenarios, we engage 7 practitioners from distinct domains to review and refine the data within their areas of expertise. In this way, our data quality is validated from multiple perspectives.

2.2 Dataset Statistics

DocBench comprises a total of 229 PDF documents sourced from publicly accessible online repositories along with 1,102 questions, spanning across 5 domains: Academia, Finance, Government, Law, and News. As shown in Table 1, we conduct comprehensive statistical analysis across various angles, encompassing the number of questions, documents, and average token counts within each. Given the unique nature of our task input, which involves processing PDF files, we additionally include information such as page count and file size. Figure 3 shows data distribution in DocBench based on various classification criteria.

2.2.1 QA-pair Type

The types of QA pairs can be mainly divided into four groups: *text-only* (37.4%), *multimodal* (27.9%), *meta-data* (23.4%), and *unanswerable* (11.3%). The *text-only* and *multimodal* types collectively account for over half (65.3%), centering on the abilities to comprehend long contexts and interpret information from different modalities. Besides, we incorporate approximately one-third (34.7%) of questions to more closely fit the actual scenarios as well as assess the robustness of the document reading systems, including 23.4% inquiring about metadata (e.g., page numbers, word counts) and 11.3% that cannot be answered based on the given document.

2.2.2 Question Type

The types of questions can be primarily separated into four categories according to the inquiry focus: *what / who / where / when / which* (58.6%), *Y/N* (22.1%), *how* (18.8%), and *why* (0.5%). These categories respectively need specific information or details, straightforward *yes* or *no* responses, methods or degrees, and the underlying reasons behind actions or phenomena. Figure 3(b) provides a detailed data distribution based on question types.

Sources	# Correct / Wrong by different evaluators				Agreement (human and automatic evaluators)		
	Human	GPT-4	GPT-3.5	StrMatch	GPT-4	GPT-3.5	StrMatch
KimiChat	24 / 16	23 / 17	33 / 7	0 / 40	97.5%	75.0%	40.0%
Qwen-2.5	17 / 23	18 / 22	31 / 9	0 / 40	97.5%	57.5%	57.5%
Gemma (7B)	19 / 21	18 / 22	18 / 22	0 / 40	97.5%	75.0%	52.5%
Mixtral (7B)	14 / 26	14 / 26	26 / 14	0 / 40	100.0%	65.0%	65.0%
Llama-3 (70B)	16 / 24	15 / 25	28 / 12	0 / 40	97.5%	62.5%	60.0%
Total	90 / 110	88 / 112	136 / 64	0 / 200	98.0%	67.0%	55.0%

Table 2: The GPT-4 automatic evaluator shows a 98% agreement with human annotators. We randomly sample 40 questions and answers from five systems, asking human annotators to assess their accuracy. We then employ string matching (StrMatch), GPT-3.5, and GPT-4 as automatic evaluators. Finally, we measure the agreement between the human and these automatic evaluators.

The interrogative *what* holds a dominant proportion at 40.8%, which is reasonable as users commonly seek precise information in a document.

2.2.3 Answer Type

The types of answers can be partitioned into four classes: *numerical* (37.4%), *textual* (35.7%), *boolean* (17.3%), and *others* (9.6%). Within the *numerical* class, 69% originate from the domains of *academia* and *finance*, as these documents naturally require extensive use of numbers to convey information, such as performance metrics in academic papers and figures in financial reports.

2.3 Evaluation Setup

Evaluation Process Our dataset diversity poses two major evaluation challenges: (i) The evaluation methods vary depending on the answer type. For example, for boolean or numerical answers, a fair evaluator only needs to verify the correctness of a binary *yes/no* response or a specific number using simple techniques like string matching or number extraction. In contrast, textual responses require more nuanced standards such as natural language generation (NLG) metrics. Thus, accurately determining the appropriate evaluation method becomes complex when the answer type is unknown. (ii) Different LLMs and systems exhibit substantial variations in the organization and style of their outputs, potentially leading to biases in traditional evaluation approaches. Therefore, we capitalize on the prowess of LLMs that have proven to be decent evaluators and can be easily adapted to the assessment of various answer types (Fu et al., 2023; Liu et al., 2023; Wang et al., 2023). Inspired by Liu et al. (2023), we clearly define the evaluation criteria for various types within the instruction prompt and then instruct GPT-4 to assign a score of 0 (incorrect) or 1 (correct). After evaluating 200 examples by both human evaluators and GPT-4, we found that the GPT-4 automatic evaluator shows a 98% agreement with human annotators, significantly exceeding the traditional string matching approach. Details of this experiment is shown in Table 2, and details of evaluation instruction prompts are attached in Appendix B.

Metrics As mentioned above, we instruct GPT-4 to assign a score of 0 (incorrect) or 1 (correct), thus using Accuracy (abbreviated as Acc.) to measure system performance. We report accuracy across all instances, as well as for each domain and QA-pair type in Table 3.

3 Experiments and Analysis

3.1 Experimental Setup

We conduct a comprehensive evaluation of 22 LLM-based document reading systems, encompassing both proprietary systems that support document uploads and a series of *parse-then-read* pipelines. For *parse-then-read* pipelines, we leverage the `fitz` package to extract text and image blocks from `PDF` files. We retain the original texts and line breaks for text chunks while we denote the i-th image as *[image i]* for images. Our selection for the proprietary systems includes GPT-4 and GPT-4o (Achiam et al., 2023) from OpenAI, GLM-4 [9] from ZhipuAI, Kimi [10] from Moonshot AI, Claude-3 [11] from Anthropic, Qwen-2.5 [12] from Alibaba Cloud, and ERNIE-3.5 [13] from Baidu. In the case of the *parse-then-read* pipelines, we assess 15 prominent LLMs as base models, featuring those from the GPT (Achiam et al., 2023; OpenAI, 2022), Llama (Touvron et al., 2023), Mistral (Jiang et al., 2024), Yi (Young et al., 2024),

[9] `https://chatglm.cn/main/doc`
[10] `https://kimi.moonshot.cn`
[11] `https://claude.ai/chats`
[12] `https://tongyi.aliyun.com/qianwen`
[13] `https://yiyan.baidu.com`

Methods	Form	Ver. /Size	File /Cxt.	Domain					Type				Overall Acc.
				Aca.	Fin.	Gov.	Laws	News	Text.	Multi.	Meta.	Una.	
Human	-	-	-	83.0	82.2	77.8	75.0	86.4	81.4	83.3	77.5	82.2	81.2
LLM-based systems													
GPT-4	*API*	0409	100M	65.7	**65.3**	75.7	69.6	79.6	**87.9**	**74.7**	50.8	37.1	69.8
GPT-4o	*API*	0513	100M	56.4	56.3	73.0	65.5	75.0	85.0	62.7	50.4	17.7	63.1
GLM-4	*Web*	-	20M	55.8	35.4	61.5	62.8	82.0	73.1	50.3	48.8	33.1	56.5
KimiChat	*Web*	-	100M	62.4	61.8	**77.0**	78.5	**87.2**	87.6	65.3	50.4	**71.8**	**70.9**
Claude-3	*Web*	Opus	10M	**73.9**	40.6	70.3	**79.1**	86.6	80.8	64.6	**54.3**	58.9	67.6
Gemini-1.5	*Web*	Pro	30M	60.4	42.5	57.4	71.7	74.3	74.0	30.8	53.8	60.2	55.4
Qwen-2.5	*Web*	-	150M	42.9	29.9	51.4	55.5	69.2	61.7	31.8	36.0	58.1	46.9
ERNIE-3.5	*Web*	-	10M	56.4	37.5	54.7	58.1	58.1	63.6	47.7	36.8	54.0	51.8
Parse-then-Read Pipelines													
GPT-4	*API*	0409	128k	**70.0**	**47.9**	**68.9**	**70.7**	**93.6**	**79.1**	**63.3**	**54.3**	70.2	**67.9**
GPT-3.5	*API*	0125	16k	49.8	24.0	58.8	50.3	83.7	65.0	37.0	42.6	44.4	49.6
ChatGLM3	*Open*	6B	128k	34.7	41.7	58.1	51.3	58.1	70.4	40.3	31.0	12.1	46.2
Gemma	*Open*	7B	8k	34.3	12.5	43.2	34.0	65.1	43.0	17.2	21.3	**77.4**	34.6
Mixtral	*Open*	7B	32k	42.6	29.2	58.8	50.3	82.0	71.8	33.8	38.4	30.6	48.7
InternLM2	*Open*	7B	32k	38.6	27.1	52.0	46.1	65.7	63.3	28.9	35.3	25.8	42.9
Llama-3	*Open*	8B	8k	44.6	23.6	61.5	54.5	86.6	68.0	29.2	45.0	49.2	49.6
Yi-1.5	*Open*	9B	16k	40.6	26.4	58.1	52.4	83.1	66.0	33.8	45.7	27.4	47.9
Llama-2	*Open*	13B	4k	20.8	18.4	29.7	23.6	55.2	43.4	15.9	21.7	12.9	27.2
Phi-3	*Open*	14B	128k	50.2	44.4	65.5	64.4	76.7	77.4	45.8	45.3	44.4	57.4
InternLM2	*Open*	20B	32k	43.2	28.5	59.5	54.5	80.8	73.3	33.4	43.0	22.6	49.4
Yi-1.5	*Open*	34B	16k	47.2	27.1	59.5	56.5	78.5	68.2	39.0	49.2	19.4	50.1
Command-R	*Open*	35B	128k	49.5	38.9	66.2	64.4	80.8	78.4	50.0	49.6	13.7	56.4
Mixtral-8x7B	*Open*	47B	32k	48.5	31.9	60.1	59.2	81.4	76.0	42.9	46.9	12.1	52.7
Llama-3	*Open*	70B	8k	52.1	25.3	68.2	59.2	90.7	69.2	38.6	49.2	56.5	54.5

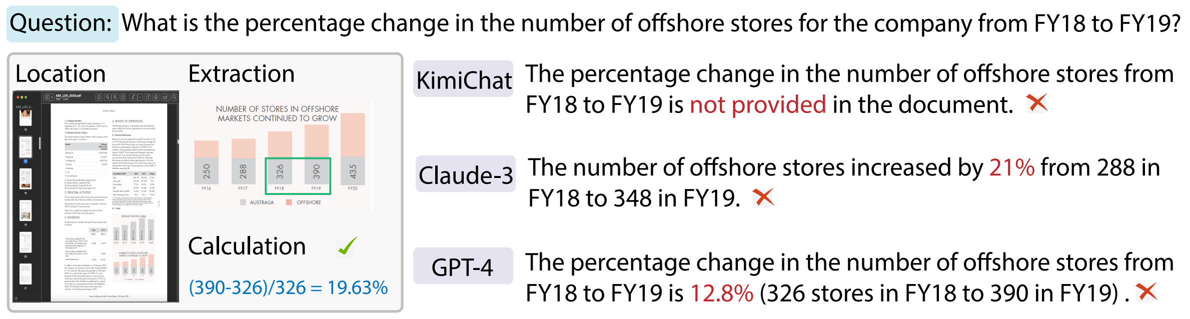
Table 3: Results on DocBench across various types and domains. *Ver./Size* stands for the model version or size; *File* denotes the maximum uploaded file size; *Cxt.* refers to model's context length.

InternLM (Cai et al., 2024), Phi-3 (Abdin et al., 2024), Gemma (Team et al., 2024), ChatGLM3 (Du et al., 2021), and Command-R (CohereAI, 2024) families. The selection of base open-sourced LLMs adheres to three guiding principles: (i) official release with *instruct* or *chat* versions that are supported by vLLM (Kwon et al., 2023) framework; (ii) model sizes ranging from 7B to 70B to accommodate GPU memory constraints; (iii) availability of the longest context length and the latest version.

3.2 Results and Discussion

Table 3 showcases the performance of various document reading systems on DocBench. Our findings reveal substantial variations in document reading capabilities among these systems, driven by differences in their foundational models, context length limitations, diverse design and implementation approaches, and etc. In this section, we will provide further discussions to delve deeper into the pros and cons of existing systems, as well as uncover the core challenges posed by DocBench.

1. Interpreting multi-modal and metadata information. Figure 4 presents a case study illustrating the unique challenge of answering multi-modal questions in DocBench. We observe that leading proprietary LLM-based systems often fail due to errors in one of the steps in the *Location→Extraction→Calculation* sequence. Take the case study as an example, in the first step, KimiChat fails to locate the relevant chart on page 17. In the extraction phase, Claude-3 misidentifies the data as *288 & 348*, instead of the correct *326 & 390*. Finally, while GPT-4 locates and extracts the correct information, it errs in calculating the percentage change, demonstrating the complexity of these questions. Interestingly, parse-then-read pipelines can achieve reasonable performance on multi-modal questions (e.g., 63.3% for GPT-4).



Question: What is the percentage change in the number of offshore stores for the company from FY18 to FY19?

KimiChat: The percentage change in the number of offshore stores from FY18 to FY19 is not provided in the document. ✗

Claude-3: The number of offshore stores increased by 21% from 288 in FY18 to 348 in FY19. ✗

GPT-4: The percentage change in the number of offshore stores from FY18 to FY19 is 12.8% (326 stores in FY18 to 390 in FY19) . ✗

Figure 4: To address multi-modal questions in DocBench, it is essential to: (i) identify the relevant figure/table (Location); (ii) extract specific data (Extraction); (iii) perform necessary calculations (Calculation). In this case study, KimiChat fails to locate the figure, Claude-3 retrieves incorrect data, and GPT-4, despite succeeding in the first two steps, struggles with the calculation.

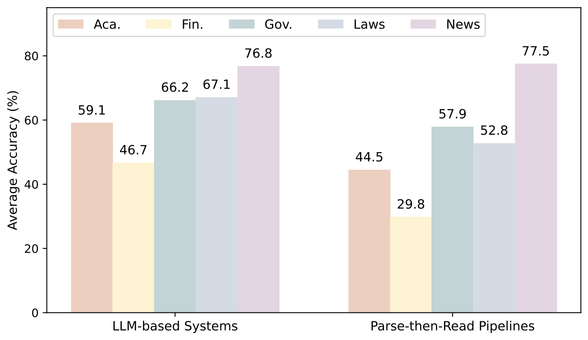


Figure 5: Average accuracy (%) of two methods under five different domains.

This is likely because the parsing process captures certain table information, and documents often include textual descriptions of figures. Meanwhile, for metadata-related questions, current methods generally lack attention to global information, resulting in relative low performances (below 55%).

2. Handling lengthy documents. Handling lengthy documents is demanding, especially in real-world scenarios where document size can be virtually unlimited. Proprietary LLM-based systems struggle with uploading extensive files, while the parse-then-read pipelines with open-sourced LLMs are constrained by their maximum context length, leading to varying degrees of information loss. As shown in Figure 5, both methods perform poorly in the finance domain but achieve higher performance in the news domain. This discrepancy arises because financial documents are typically longer and contain richer information, whereas news files are limited to single front pages with fewer messages. Furthermore, certain strong models with relatively short context lengths may excel with smaller files, but context length becomes a crucial factor when it comes to large files. For instance, the *8k* Llama-3 family performs exceptionally well in the news domain, but is outperformed by all the *128k* models in the finance domain. Besides, we discover that KimiChat and Command-R, which are specifically enhanced for long-context and Retrieval-Augmented Generation (RAG) capabilities, achieve decent results on *text-only* questions. Therefore, a key challenge lies in adapting these systems to handle documents of varying lengths while balancing the foundational model's capabilities and context length constraints.

3. Faithfulness to user-provided documents Most existing document reading systems falter when faced with unanswerable questions based on the provided document, exhibiting a lack of fidelity. Remarkably, Gemma and KimiChat perform better in such scenarios, which represents a crucial capability since users often expect systems to answer questions strictly based on given files. Intriguingly, despite the commonly-shared base model on GPT-4, there is a notable performance gap between the system and the parse-then-read pipeline in handling unanswerable questions (i.e., 37.1% and 70.2 % for system and pipeline, respectively). We analyze that this may be due to: (i) the proprietary LLM-based system have undergone optimizations on the base model, potentially causing overfitting; (ii) GPT-4 tends to adhere more closely to the in-context learning information. Such phenomenon thus underscores a critical challenge for future document reading systems on enhancing fidelity to the given documents.

4 Related Works

4.1 Recent Advances of LLMs and LLM-based Systems

The latest generation of LLMs, such as GPT-4 (Achiam et al., 2023), Llama-3 (Touvron et al.,

2023) and Claude-3 (Anthropic, 2024), have significantly extended the capabilities of language models (Zhao et al., 2023; Chang et al., 2024; Wang et al., 2024a). These models are pre-trained on vast amounts of web-scale data, enabling them to perform a wide range of human-instructed tasks with impressive performance. Despite their remarkable performance, standalone LLMs may not be sufficient for many real-world applications. For example, LLMs lack access to real-time information and may struggle with tasks that require up-to-date knowledge (Vu et al., 2023). Moreover, real-world applications often require non-text inputs parsing, code execution, API calling and interaction with external environments (Lee et al., 2024; Labs, 2024; Jimenez et al., 2023; Zhou et al., 2023; Xie et al., 2024; Guo et al., 2024). The overall task completion usually requires multiple reasoning, execution and reflection steps that cannot be accomplished in a simple input-output manner (Yao et al., 2023; Shinn et al., 2023; Wang et al., 2024b). To overcome the limitations of standalone LLMs, recent efforts have incorporated additional components and sophisticated system design. These systems, such as Microsoft's Co-Pilot[14] and OpenAI's GPT-4 all-in-one[15], aim to provide more comprehensive and practical solutions for real-world applications. Other pioneering efforts on designing LLM-based systems include web agents (Zheng et al., 2024; He et al., 2024; Ma et al., 2023), software agents (Yang et al., 2024; Labs, 2024) and computer agents (Wu et al., 2024) that can interact with external resources (e.g., websites, search engine, code repositories or computers) and perform multi-step tasks. The success of these systems relies on integrating powerful LLMs with well-designed architectures and components that enable them to handle complex tasks effectively.

4.2 Document reading: Datasets and Methods

Document reading is a critical area where LLM-based systems have demonstrated significant advancements. Proprietary developers such as OpenAI[16] and Anthropic[17] have introduced advanced systems that can take a user-provided document as input, parse its structure, extract relevant metadata, and handle long texts and images to provide accurate responses. While these systems build upon the fundamental capabilities of their underlying LLMs (Zeng et al., 2022; Bai et al., 2023; Achiam et al., 2023; Anthropic, 2024), they differ in their design and implementation, with some systems excelling in long-context reading and others focusing on retrieval-augmented methods to improve document reading ability. Despite claims of effectiveness and efficiency in online public blogs, the absence of a standardized benchmark makes it difficult to objectively evaluate and compare the document reading performance across these systems. Existing benchmarks relevant to document reading are unable to adequately reflect the real performance of these systems. Datasets focusing on document understanding such as Doc2Dial (Feng et al., 2020), ConditionalQA (Sun et al., 2022) and those specifically focusing on long-context reading like NarrativeQA (Kočiský et al., 2018) and QuALITY (Pang et al., 2022), primarily use text as input only, ignoring the complex nature of document structure and multi-modal information. On the other hand, multi-modal document reading datasets like DocVQA (Mathew et al., 2021), ChartQA (Masry et al., 2022), OCR-VQA (Mishra et al., 2019), and InfoVQA (Mathew et al., 2022) include multi-modal inputs and preserve the original document structure and layout. However these datasets often capture only parts of document (e.g. tables or figures) and ignored substantial amount of textual content. However, DocBench requires systems to process the full documents as intact files and covers different types of questions targeting various abilities, which can more accurately evaluate the capabilities of LLM-based document reading systems in real-world scenarios.

5 Conclusion

In this paper, we introduce DocBench, a novel benchmark created to assess LLM-based document reading systems in a comprehensive and fine-grained manner. DocBench consists of 229 documents and 1,102 questions, spanning 5 domains and 4 question types, developed with the help of human annotators and synthetic questions. We evaluate both proprietary LLM systems, accessible via web interfaces or APIs, and a parse-then-read approach using open-source LLMs. Our findings reveal significant disparities in document reading capabilities among these systems, highlighting current limitations, presenting potential challenges, and thus driving forward progress in this field.

[14] https://copilot.microsoft.com
[15] https://chat.openai.com
[16] OpenAI's ChatGPT: https://chat.openai.com
[17] Anthropic's Claude: https://claude.ai/chats

6 Limitation

While DocBench aims to encompass a wide range of real-world document-related questions, it is not exhaustive. Our benchmark primarily focuses on the four most common question types, leaving other potential types unaddressed. Additionally, our evaluation of proprietary LLM-based document reading systems is limited. Many of these systems, such as OpenAI-o1, are accessible only through web interfaces with restricted access and lack APIs, which makes the evaluation process slow and challenging.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Claude 3 haiku: our fastest model yet.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Abeba Birhane, Atoosa Kasirzadeh, David Leslie, and Sandra Wachter. 2023. Science in the age of large language models. *Nature Reviews Physics*, 5(5):277–280.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

CohereAI. 2024. Introducing command r.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*.

Song Feng, Hui Wan, Chulaka Gunasekara, Siva Patel, Sachindra Joshi, and Luis Lastras. 2020. doc2dial: A goal-oriented document-grounded dialogue dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8118–8128, Online. Association for Computational Linguistics.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *Preprint*, arXiv:2302.04166.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. Ds-agent: Automated data science by empowering large language models with case-based reasoning. *Preprint*, arXiv:2402.17453.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Cognition Labs. 2024. Devin, ai software engineer.

Jinqi Lai, Wensheng Gan, Jiayang Wu, Zhenlian Qi, and Philip S Yu. 2023. Large language models in law: A survey. *arXiv preprint arXiv:2312.03718*.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. *arXiv preprint arXiv:2402.09727*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. 2023. Laser: Llm agent with state-space exploration for web navigation. *Preprint*, arXiv:2309.08172.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. 2022. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1697–1706.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. 2019. Ocr-vqa: Visual question answering by reading text in images. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 947–952. IEEE.

OpenAI. 2022. Introducing chatgpt.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. 2022. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Haitian Sun, William Cohen, and Ruslan Salakhutdinov. 2022. ConditionalQA: A complex reading comprehension dataset with conditional answers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3627–3637, Dublin, Ireland. Association for Computational Linguistics.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, et al. 2023. Freshllms: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023. Is ChatGPT a good NLG evaluator? a preliminary study. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, pages 1–11, Singapore. Association for Computational Linguistics.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):1–26.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better llm agents. In *ICML*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024. Os-copilot: Towards generalist computer agents with self-improvement. *Preprint*, arXiv:2402.07456.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded. *Preprint*, arXiv:2401.01614.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Annotation Process

Since the QA-pair generation process requires data annotators to deeply understand the motivations behind our benchmark construction, and considering the initial training costs and the need to manually annotate about 350 QA pairs, we've decided to assign 2 annotators to this task.

The annotation process presents as follows:

- We first communicate the motivation behind our work to the annotators and explain the concepts of meta-data and unanswerable questions in detail.
- Next, we provide 10 example QA pairs for reference (5 for each type).
- Finally, each annotator generates 170 QA pairs. They then exchange their annotations for double-checking and review.

B Instruction Prompts

B.1 Response Evaluation

Detailed instruction prompts for response evaluation are shown in Table 4.

B.2 QA-pair Generation

Details of instruction prompts for generating QA pairs are attached in Table 5. We discover that simply passing diagrams to GPT-4V leads to subpar question quality. This issue likely stems from the fact that figures or tables without accompanying text descriptions typically lack sufficient information, thus causing the generated QA pairs to deviate from their intended meanings. In addition, we observe that adding difficulty settings for QA generation (e.g., *Easy*, *Medium*, *Hard*) in the instruction prompt can result in higher quality. We analyze that this may be due to the model being able to favor higher generation quality in potential comparisons.

C Performance Comparison

Figure 6 demonstrates the relative performance of LLM-based systems and parse-then-read pipelines against the best on DocBench. For LLM-based systems, KimiChat consistently scores high across various metrics, demonstrating balanced performance. Notably, GPT-4 performs poorly in the unanswerable category, indicating potential overfitting in optimized GPT-4 file systems, which leads to decreased fidelity to given documents. Additionally, Claude-3 excels in the meta-data category, highlighting its superior ability to comprehend high-level metadata information. For parse-then-read pipelines, we select models with the highest overall accuracy for comparison. Unlike LLM-based systems, GPT-4 demonstrates consistently high and balanced performance across all aspects within this pipeline. Notably, significant discrepancies arise in handling multi-modal and unanswerable questions, where GPT-4 and Gemma exhibit clear distinctions from the remaining methods.

D Analysis of Input Sources

Table 7 presents the impact of different input sources on model performance. We provide questions to GPT-4 and GPT-4o, both with and without attached files. Remarkably, even without files, the models correctly answer a portion of the questions (19.1% for GPT-4 and 21.7% for GPT-4o). Our analysis reveals that the correctly answered questions are predominantly textual and are largely associated with government, law, and news domains. This trend suggests that the models' underlying training data is heavily skewed towards these categories, enabling them to answer some questions accurately without additional files. Moreover, as GPT-4o is an optimized version of GPT-4, it likely benefits from a broader and more training data.

Table 4: Instruction Prompts in Response Evaluation.

`System Content:`

You are a helpful evaluator.

`Prompt:`

Task Overview:

You are tasked with evaluating user answers based on a given question, reference answer, and additional reference text. Your goal is to assess the correctness of the user answer using a specific metric.

Evaluation Criteria:

1. Yes/No Questions: Verify if the user's answer aligns with the reference answer in terms of a "yes" or "no" response.
2. Short Answers/Directives: Ensure key details such as numbers, specific nouns/verbs, and dates match those in the reference answer.
3. Abstractive/Long Answers: The user's answer can differ in wording but must convey the same meaning and contain the same key information as the reference answer to be considered correct.

Evaluation Process:

1. Identify the type of question presented.
2. Apply the relevant criteria from the Evaluation Criteria.
3. Compare the user's answer against the reference answer accordingly.
4. Consult the reference text for clarification when needed.
5. Score the answer with a binary label 0 or 1, where 0 denotes wrong and 1 denotes correct.

NOTE that if the user answer is 0 or an empty string, it should get a 0 score.

Question: `{{question}}`
User Answer: `{{sys_ans}}`
Reference Answer: `{{ref_ans}}`
Reference Text: `{{ref_text}}`

Evaluation Form (score ONLY):

- Correctness:

Table 5: Instruction Prompts in QA-pair Generation.

`System Content:`

You are a helpful assistant that can generate question-answer pairs.

`Text-only QA:`

Based on the above text, please design three question-answer pairs with different levels of difficulty: Easy, Medium, Hard.

The questions should be close-ended and should be answered based on the provided text.

The answer form should be as diverse as possible, including [Yes/No, Short Answer, Long Answer, Abstractive Answer].

You should provide the reference in the text and the answer form if possible.

The output should be formalized as: "'Q: | A: | Reference: | Difficulty Level: | Answer Form:"'

`Multimodal QA (w/table+text):`

Based on the above table and text, please design three question-answer pairs with different levels of difficulty: Easy, Medium, Hard.

The text provided is text related to the table, which can provide more reference for question generation, but the focus is still on the table itself.

These questions require locating the specific information, simple or complex calculations, comparisons, finding the maximum and minimum, reading across rows and columns, etc.

Note that these questions also need to be realistic. You should provide the reason if possible.

The output should be formalized as: "'Q: | A: | Reference: | Difficulty Level: | Answer Form:"'

`Multimodal QA (w/figure+text):`

Based on the above figure and text, please design three question-answer pairs with different levels of difficulty: Easy, Medium, Hard.

The text provided is text related to the figure, which can provide more reference for question generation, but the focus is still on the figure itself.

These questions require a deep reading of the meaning of the image.

Note that these questions also need to be realistic. You should provide the reason if possible.

The output should be formalized as: "'Q: | A: | Reason: | Difficulty Level: | "'

`Multimodal QA (w/table):`

Based on the above image, please design three question-answer pairs with different levels of difficulty: Easy, Medium, Hard.

These questions require locating the specific information, simple or complex calculations, comparisons, finding the maximum and minimum, reading across rows and columns, etc.

Note that these questions also need to be realistic. You should provide the reason if possible.

The output should be formalized as: "'Q: | A: | Reason: | Difficulty Level: | "'

`Multimodal QA (w/figure):`

Based on the above image, please design three question-answer pairs with different levels of difficulty: Easy, Medium, Hard.

These questions require a deep reading of the meaning of the image. Note that these questions also need to be realistic. You should provide the reason if possible.

The output should be formalized as: "'Q: | A: | Reason: | Difficulty Level: | "'

Table 6: Examples of instances from DOCBENCH, with multiple labels indicating our data diversity.

Question	Answer	Labels	Document
Why does the model not perform as well in German compared to Spanish and Dutch?	Due to its **complex morphology and compound words**...	`<Aca.><Why>` `<Text-only>` `<Textual>`	When and Why are Pre-trained Word Embeddings Useful for Machine Translation [clickable file link]
By **how much** did the number of Erica users increase from 2018 to 2019?	The number increased by **5.5 million**...	`<Fin.><How>` `<Multimodal>` `<Numerical>`	Bank of America Annual Report 2020 [clickable file link]
What is the primary focus of Bureau Objective 3.4?	The report **does not contain** such objective.	`<Gov.> <Wh->` `<Unanswerable>` `<Others>`	Governmental report from *Secretary's Office of Global Women's Issues* 2022 [clickable file link]
How many times does the report mention "scientific ethics"?	The report mentions "scientific ethics" **11** times.	`<Laws><How>` `<Meta-data>` `<Numerical>`	Report on *Regulation of Stem Cell Research* from Library of Congress 2023 [clickable file link]
Is the article about Hurricane Ian's impact in Florida written by multiple authors?	**Yes**, the article is about Hurrican Ian's impace in Florida...	`<News><Y/N>` `<Meta-data>` `<Boolean>`	New York Times front page on `2022-09-30` [clickable file link]

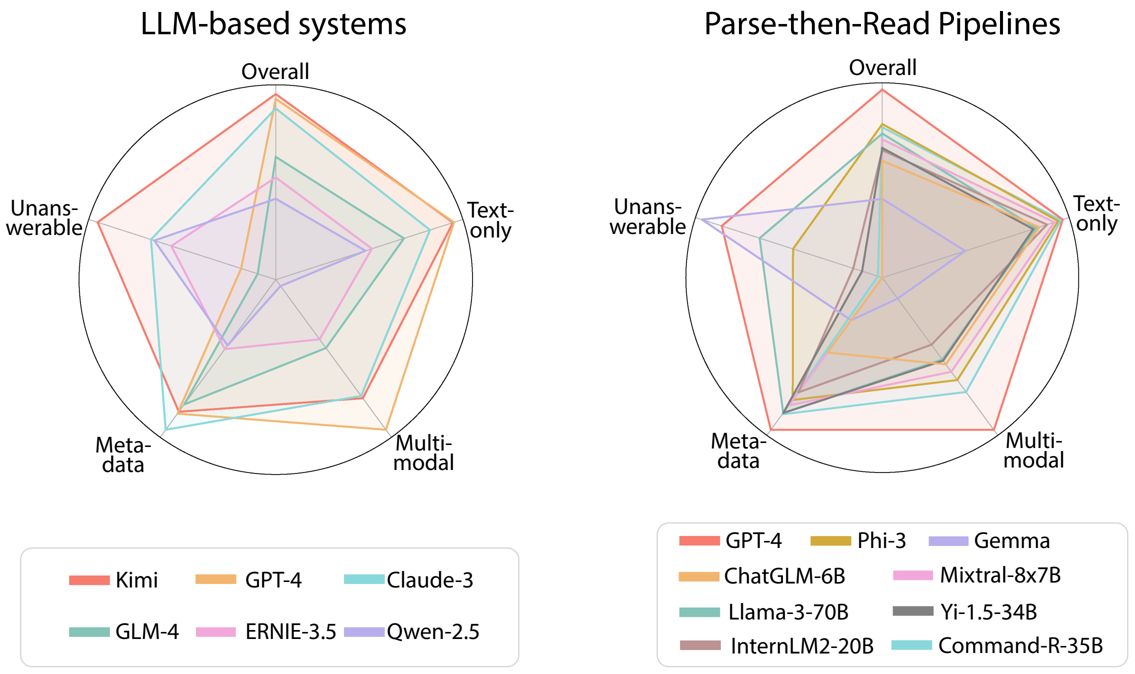


Figure 6: Performance (Relative) of two major methods on DOCBENCH against the best.

Table 7: Analyzing the Influence of Input Sources: We deliver questions with attached files and without files to GPT-4 and GPT-4o for evaluation, respectively.

Methods	**Domain**					**Type**				**Overall** Acc.
	Aca.	**Fin.**	**Gov.**	**Laws**	**News**	**Text.**	**Multi.**	**Meta.**	**Una.**	
GPT-4										
w/ file	65.7	65.3	75.7	69.6	79.6	87.9	74.7	50.8	37.1	69.8
w/o file	10.9	10.8	23.0	29.3	32.6	40.8	8.1	1.6	10.5	19.1
GPT-4o										
w/ file	56.4	56.3	73.0	65.5	75.0	85.0	62.7	50.4	17.7	63.1
w/o file	11.2	13.5	29.1	31.9	36.0	46.6	10.7	2.3	6.5	21.7