PDFTriage: Question Answering over Long, Structured Documents

Jon Saad-Falcon
Stanford University
jonsaadfalcon@stanford.edu

Joe Barrow
Adobe Research
jbarrow@adobe.com

Alexa Siu
Adobe Research
asiu@adobe.com

Ani Nenkova
Adobe Research
nenkova@adobe.com

Ryan A. Rossi
Adobe Research
ryrossi@adobe.com

Franck Dernoncourt
Adobe Research
dernonco@adobe.com

Abstract

Large Language Models (LLMs) have issues with document question answering (QA) in situations where the document is unable to fit in the small context length of an LLM. To overcome this issue, most existing works focus on retrieving the relevant context from the document, representing them as plain text. However, documents such as PDFs, web pages, and presentations are naturally structured with different pages, tables, sections, and so on. Representing such structured documents as plain text is incongruous with the user's mental model of these documents with rich structure. When a system has to query the document for context, this incongruity is brought to the fore, and seemingly trivial questions can trip up the QA system. To bridge this fundamental gap in handling structured documents, we propose an approach called *PDFTriage* that enables models to retrieve the context based on either structure or content. Our experiments demonstrate the effectiveness of the proposed *PDFTriage-augmented* models across several classes of questions where existing retrieval-augmented LLMs fail. To facilitate further research on this fundamental problem, we release our benchmark dataset consisting of 900+ human-generated questions over 80 structured documents from 10 different categories of question types for document QA.

1 Introduction

When a document does not fit in the limited context window of an LLM, different strategies can be deployed to fetch relevant context. Current approaches often rely on a pre-retrieval step to fetch the relevant context from documents (Pereira et al., 2023; Gao et al., 2022). These pre-retrieval steps tend to represent the document as plain text chunks, sharing some similarity with the user query and potentially containing the answer. However, many document types have rich structure, such as web pages, PDFs, presentations, and so on. For these structured documents, representing the document as plain text is often incongruous with the user's mental model of a *structured document*. This can lead to questions that, to users, may be trivially answerable, but fail with common/current approaches to document QA using LLMs. For instance, consider the following two questions:

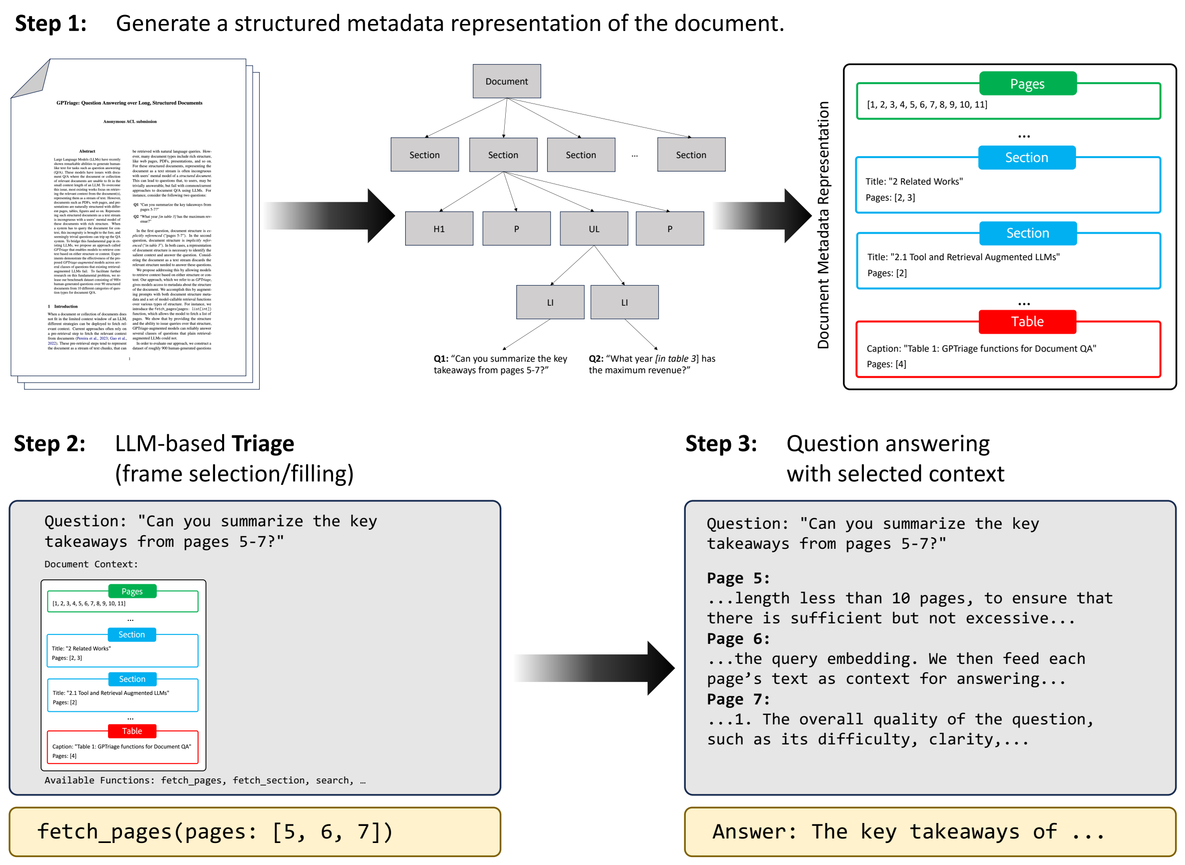
Q1 "Can you summarize the key takeaways from pages 5-7?"

Q2 "What year *[in table 3]* has the maximum revenue?"

In the first question, document structure is *explicitly referenced* ("pages 5-7"). In the second question, document structure is *implicitly referenced* ("*in table 3*"). In both cases, a representation of document structure is necessary to identify the salient context and answer the question. Considering the document as plain text discards the relevant structure needed to answer these questions.

We propose addressing this simplification of documents by allowing models to retrieve the context based on either structure or content. Our approach, which we refer to as *PDFTriage*, gives models access to metadata about the structure of the document. We leverage document structure by augmenting prompts with both document structure metadata and a set of model-callable retrieval functions over various types of structure. For example, we introduce the `fetch_pages(pages: list[int])` function, which allows the model to fetch a list of pages. We show that by providing the structure and the ability to issue queries over that structure, PDFTriage-augmented models can reliably answer several classes of questions that plain retrieval-augmented LLMs could not.

In order to evaluate our approach, we construct a dataset of roughly 900 human-written questions over 90 documents, representing 10 different categories of questions that users might ask. Those


Step 1: Generate a structured metadata representation of the document.
GPTriage: Question Answering over Long, Structured Documents
Document
Section
Section
Section
...
Section
H1
P
UL
P
LI
LI
Q1: "Can you summarize the key takeaways from pages 5-7?"
Q2: "What year [in table 3] has the maximum revenue?"
Document Metadata Representation
Pages
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
...
Section
Title: "2 Related Works"
Pages: [2, 3]
Section
Title: "2.1 Tool and Retrieval Augmented LLMs"
Pages: [2]
...
Table
Caption: "Table 1: GPTriage functions for Document QA"
Pages: [4]
Step 2: LLM-based Triage (frame selection/filling)
Step 3: Question answering with selected context

```
Question: "Can you summarize the key
takeaways from pages 5-7?"
Document Context:
Available Functions: fetch_pages, fetch_section, search, …
```

```
fetch_pages(pages: [5, 6, 7])
```

```
Question: "Can you summarize the key
takeaways from pages 5-7?"

Page 5:
...length less than 10 pages, to ensure that
there is sufficient but not excessive...
Page 6:
...the query embedding. We then feed each
page's text as context for answering...
Page 7:
...1. The overall quality of the question,
such as its difficulty, clarity,...
```

```
Answer: The key takeaways of ...
```

Figure 1: **Overview of the PDFTriage technique**: PDFTriage leverages a PDF's structured metadata to implement a more precise and accurate document question-answering approach. It starts by generating a structured metadata representation of the document, extracting information surrounding section text, figure captions, headers, and tables. Next, given a query, a LLM-based Triage selects the document frame needed for answering the query and retrieves it directly from the selected page, section, figure, or table. Finally, the selected context and inputted query are processed by the LLM before the generated answer is outputted.

Document Question Answering . Several datasets have been constructed to benchmark different aspects of document-focused question-answering. DocVQA (Mathew et al., 2021) is a visual question-answering dataset focused that uses document scans. A recent work by Landeghem et al. (2023) focused on a dataset for document understanding and evaluation called DUDE, which uses both scans and born-digital PDFs. Both DUDE and DocVQA have questions that can be answered short-form; DUDE answers average roughly 3.35 tokens and DocVQA tokens average 2.11 tokens. QASPER (Dasigi et al., 2021) is a dataset focused on information-seeking questions and their answers from research papers, where the documents are parsed from raw LaTeXsources and the questions are primarily focused on document contents. The PDFTriage evaluation dataset seeks to expand on the question types in these datasets, getting questions that can reference the document structure or content, can be extractive or abstractive, and can require long-form answers or rewrites.

3 PDFTriage: Structured Retrieval from Document Metadata

The *PDFTriage* approach consists of three steps to answer a user's question, shown in Figure 1:

1. **Generate document metadata (Sec. 3.1):** Extract the structural elements of a document and convert them into readable metadata.
2. **LLM-based triage (Sec. 3.2):** Query the LLM to select the precise content (pages, sections, retrieved content) from the document.
3. **Answer using retrieved content (Sec. 3.3):** Based on the question and retrieved content, generate an answer.

# of Documents	82
# of Questions	**908**
Easy Questions	393
Medium Questions	144
Hard Questions	266
"Unsure" Questions	105

Table 1: Dataset statistics for the PDFTriage evaluation dataset.

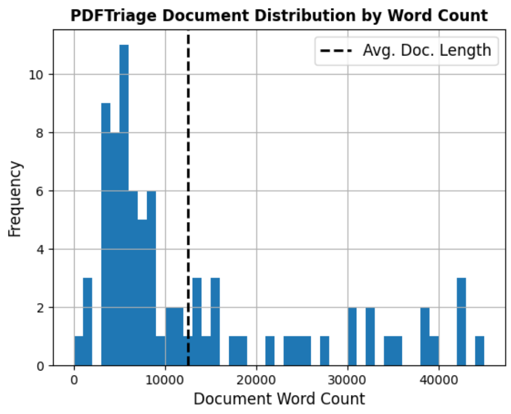


Figure 2: PDFTriage Document Distribution by Word Count

3.1 Document Representation

We consider *born-digital PDF documents* as the structured documents that users will be interacting with. Using the Adobe Extract API, we convert the PDFs into an HTML-like tree, which allows us to extract sections, section titles, page information, tables, and figures.[1] The Extract API generates a hierarchical tree of elements in the PDF, which includes section titles, tables, figures, paragraphs, and more. Each element contains metadata, such as its page and location. We can parse that tree to identify sections, section-levels, and headings, gather all the text on a certain page, or get the text around figures and tables. We map that structured information into a JSON type, that we use as the initial prompt for the LLM. The content is converted to markdown. An overview of this process is shown at the top of Figure 1.

3.2 LLM Querying of Document

PDFTriage utilizes five different functions in the approach: `fetch_pages`, `fetch_sections`, `fetch_table`, `fetch_figure`, and `retrieve`. As described in Table 2, each function allows the PDFTriage system to gather precise information related to the given PDF document, centering around structured textual data in headers, subheaders, figures, tables, and section paragraphs. The functions are used in separate queries by the PDFTriage system for each question, synthesizing multiple pieces of information to arrive at the final answer. The functions are provided and called in separate chat turns via the OpenAI function calling API,[2] though it would be possible to organize the prompting in a ReAct (Yao et al., 2022) or Toolformer (Schick et al., 2023) -like way.

3.3 Question Answering

To initialize PDFTriage for question-answering, we use the system prompt format of GPT-3.5 to input the following:

> You are an expert document question answering system. You answer questions by finding relevant content in the document and answering questions based on that content.
>
> Document: `<textual metadata of document>`

Using user prompting, we then input the query with no additional formatting. Next, the PDFTriage system uses the functions established in Section 2 to query the document for any necessary information to answer the question. In each turn, PDFTriage uses a singular function to gather the needed information before processing the retrieved context. In the final turn, the model outputs an answer to the question. For all of our experiments, we use the `gpt-35-turbo-0613` model.

4 Dataset Construction

To test the efficacy of PDFTriage, we constructed a document-focused set of question-answering tasks. Each task seeks to evaluate different aspects of document question-answering, analyzing reasoning across text, tables, and figures within a document. Additionally, we wanted to create questions ranging from single-step answering on an individual document page to multi-step reasoning across the whole document.

[1] https://developer.adobe.com/document-services/apis/pdf-extract/

[2] https://platform.openai.com/docs/api-reference

Function	Description
`fetch_pages`	Get the text contained in the pages listed.
`fetch_sections`	Get the text contained in the section listed.
`fetch_figure`	Get the text contained in the figure caption listed.
`fetch_table`	Get the text contained in the table caption listed.
`retrieve`	Issue a natural language query over the document, and fetch relevant chunks.

Table 2: PDFTriage Functions for Document QA.

We collected questions using Mechanical Turk.[3] The goal of our question collection task was to collect real-world document-oriented questions for various professional settings. For our documents, we sampled 1000 documents from the common crawl to get visually-rich, professional documents from various domains, then subsampled 100 documents based on their reading level (Flesch, 1948). [4] By collecting a broad set of document-oriented questions, we built a robust set of tasks across industries for testing the PDFTriage technique.

In order to collect a diverse set of questions, we generated our taxonomy of question types and then proceeded to collect a stratified sample across the types in the taxonomy. Each category highlights a different approach to document-oriented QA, covering multi-step reasoning that is not found in many other QA datasets. We asked annotators to read a document before writing a question. They were then tasked with writing a salient question in the specified category.

For our taxonomy, we consider ten different categories along with their associated descriptions:

1. **Figure Questions** (6.5%): Ask a question about a figure in the document.
2. **Text Questions** (26.2%): Ask a question about the document.
3. **Table Reasoning** (7.4%): Ask a question about a table in the document.
4. **Structure Questions** (3.7%): Ask a question about the structure of the document.
5. **Summarization** (16.4%): Ask for a summary of parts of the document or the full document.
6. **Extraction** (21.2%): Ask for specific content to be extracted from the document.
7. **Rewrite** (5.2%): Ask for a rewrite of some text in the document.
8. **Outside Questions** (8.6%): Ask a question that can't be answered with just the document.
9. **Cross-page Tasks** (1.1%): Ask a question that needs multiple parts of the document to answer.
10. **Classification** (3.7%): Ask about the type of the document.

In total, our dataset consists of 908 questions across 82 documents. On average a document contains 4,257 tokens of text, connected to headers, subheaders, section paragraphs, captions, and more. In Figure 2, we present the document distribution by word count. We provide detailed descriptions and examples of each of the classes in the appendix.

5 Experiments

We outline the models and strategies used in our approach along with our baselines for comparison. The code and datasets for reproducing our results will be released soon.

5.1 PDFTriage

For our primary experiment, we use our PDFTriage approach to answer various questions in the selected PDF document dataset. This strategy leverages the structure of PDFs and the interactive system functions capability of GPT-3.5 to extract answers more precisely and accurately than existing naive approaches.

5.2 Retrieval Baselines

Page Retrieval . For our first baseline, we index the pages of each individual document using *text-embedding-ada-002* embeddings. Using cosine similarity, we retrieve the pages most similar to the query embedding. We then feed each page's text as context for answering the given question until we reach the context window limit for a model.

[3]`https://mturk.com`
[4]`https://commoncrawl.org/`

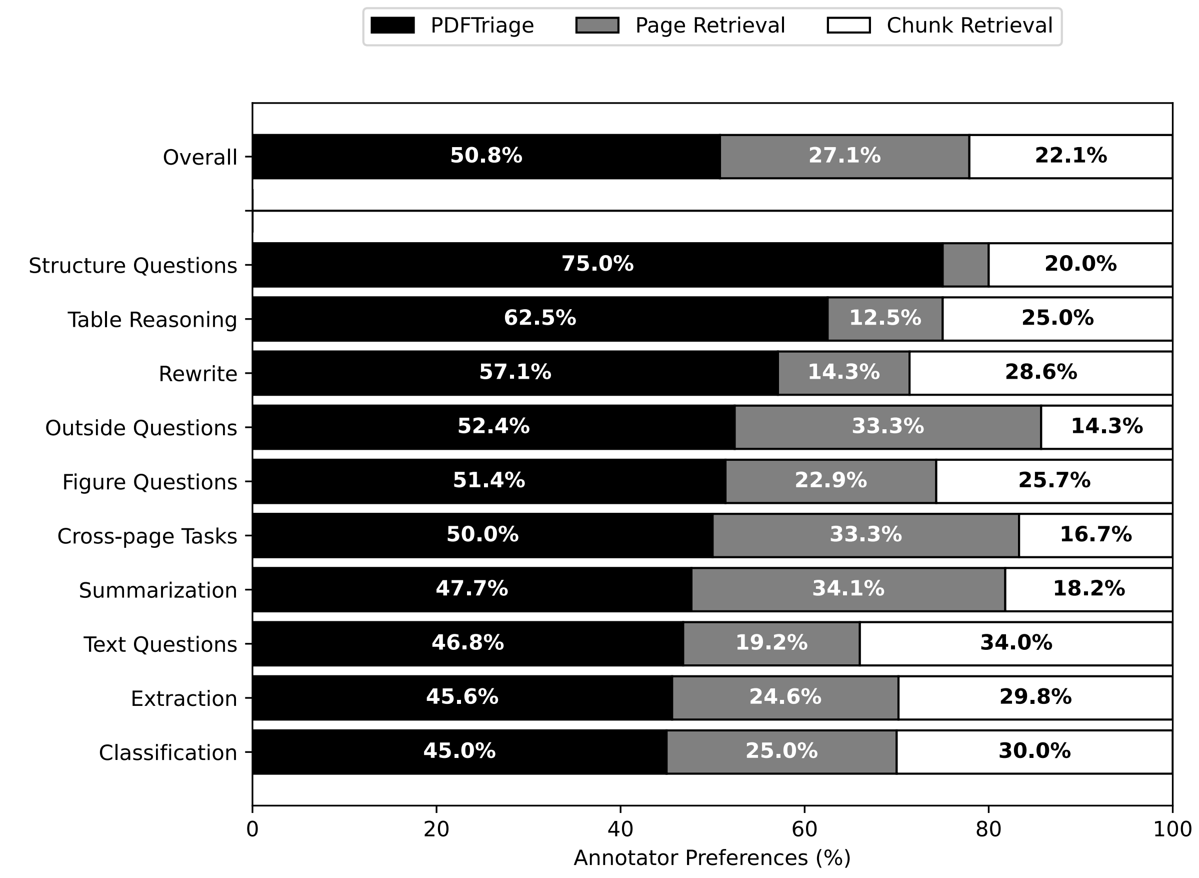


Figure 3: **User Preferences between PDFTriage and Alternate Approaches**: Overall, PDFTriage-generated answers were favored the most by the users, claiming 50.8% of the top-ranked answers overall. Furthermore, PDFTriage answers ranked higher on certain multi-page tasks, such as structure questions and table reasoning, while ranking lower on generalized textual tasks, such as classification and text questions. However, across all the question categories, PDFTriage beat both the Page Retrieval and Chunk Retrieval approaches on a head-to-head ranking.

Chunk Retrieval . In our second baseline, we concatenate all the document's text before chunking it into 100-word pieces. We then index each chunk using *text-embedding-ada-002* embeddings before using cosine similarity calculations to retrieve the chunks most similar to the query embedding. Finally, we feed each chunk's textual contents as context for answering the given question until we reach the context window limit for a model.

Prompting . For both retrieval baselines, we use the following prompt to get an answer from GPT-3.5:

> You are an expert document question answering system. You answer questions by finding relevant content in the document and answering questions based on that content.
>
> Document: `<retrieved pages/chunks>`
>
> Question: `<question>`

5.3 Human Evaluation

To measure any difference between PDFTriage and the retrieval baselines, we established a human labeling study on Upwork. In the study, we hired 12 experienced English-speaking annotators to judge the answers generated by each system. Please see Appendix A to see the full annotation questions for each question-document and its generated answers (for the overview, we use a sample question).

Our questions seek to understand several key attributes of each question-document pair as well as the associated general questions:

1. The overall quality of the question, such as its difficulty, clarity, and information needed for answering it.

2. The category of the question, using the taxon-

omy in section 4.

3. The ranking of each generated answer for the given question-document pair.

4. The accuracy, informativeness, readability/understandability, and clarity of each generated answer.

6 Results and Analysis

In Table 1, we present the annotated question difficulty of each question in our sample. Overall, the largest group of questions (43.3%) were categorized as Easy while roughly a third of questions were categorized as Hard for various reasons.

In addition to question difficulty, we asked annotators to categorize questions by type using the same categories as Section 4. Our annotation framework results in a dataset that's diverse across both question types and question difficulties, covering textual sections, tables, figures, and headings as well as single-page and multi-page querying. The diversity of questions allows us to robustly evaluate multiple styles of document-centered QA, testing the efficacy of PDFTriage for different reasoning techniques.

6.1 PDFTriage yields better answers than retrieval-based approaches.

In our annotation study, we asked the annotators to rank PDFTriage compared to our two baselines, Page Retrieval and Chunk Retrieval (Section 5). In Figure 3, we found that annotators favored the PDFTriage answer over half of the time (50.7%) and favored the Chunk Retrieval approach over the Page Retrieval approach. When comparing different provided answers for the same question, PDFTriage performs substantially better than current alternatives, ranking higher than the alternate approaches across all the question types.

6.2 PDFTriage improves answer quality, accuracy, readability, and informativeness

In our annotation study, we also asked the annotators to score PDFTriage, Page Retrieval, and Chunk Retrieval answers across five major qualities: accuracy, informativeness, readability/understandability, and clarity. We hoped to better understand the strengths of each answer for users in document question-answering tasks. In Table 3, we show that PDFTriage answers score higher than Page Retrieval and Chunk Retrieval across all answer qualities except for Clarity. Crucially, PDFTriage had the highest scores for Overall Quality and Answer Accuracy. For annotator agreement, we calculated an average Cohen's kappa score of 0.584.

	PDFTriage	*Page Retrieval*	*Chunk Retrieval*
Readability	**4.2**	4.1	4.1
Informativeness	**3.9**	3.7	3.4
Clarity	2.0	2.1	**2.3**
Accuracy	**3.8**	3.6	3.4
Overall Quality	**3.9**	3.8	3.6

Table 3: Answer Quality Scoring

In Appendix A, we provide a high-resolution breakdown of annotations for "Overall Quality" and "Accuracy" by question category. We find that PDFTriage tends to be stronger for categories like summarization, table reasoning, extraction, and figure questions which require multi-step reasoning across different parts of a document. Additionally, PDFTriage performs similarly to Page Retrieval and Chunk Retrieval on other more generalized reasoning tasks, such as text questions and classification.

6.3 PDFTriage requires fewer retrieved tokens to produce better answers

For the PDF document sample used in our study, the average token length of retrieved PDFTriage text is 1568 tokens (using the GPT-3.5 tokenizer). The average metadata length of textual inputs in document JSONs is 4,257 tokens (using the GPT-3.5 tokenizer).

While PDFTriage utilizes more tokens than Page Retrieval (3611 tokens on average) and Chunk Retrieval (3934 tokens on average), the tokens are retrieved from multiple sections of the document that are non-consecutive. Furthermore, the sections used in Page Retrieval and Chunk Retrieval are often insufficient for answering the question, as indicated by lower answer quality scores on average for "Overall Quality" and "Accuracy". However, simply concatenating all the document's text together would not ultimately replace PDFTriage due to both context window limits and the need to perform multi-hop reasoning for document QA tasks. PDFTriage helps overcome this issue through the multi-stage querying of the document, retrieving and adding context as needed for different document QA tasks.

Question	PDFTriage	Chunk Retrieval
Which file formats are listed under the “Create Using This Format” section?	Function used: `fetch_section()` The file formats listed under the "Create Using This Format" section are: CSV, JPG, LPT, PDF, PNG, PS, RTF, XLS, and XML.	The document does not provide information about the specific file formats listed under the “Create Using This Format” section.
Summarize the second paragraph in one long sentence.	Function used: `fetch_section()` The second paragraph explains the writer’s support for H.B. 5540, which aims to ban guns without serial numbers and regulate those sold in a form requiring assembly. The writer also supports the...	In her testimony, Michele Mudrick, Legislative Advocate for the Connecticut Conference United Church of Christ, expresses her support for H B 5540 and H B 5542, two acts concerning ghost guns, the permit application process, and...

Table 4: A comparison of `fetch_section()` being called successfully and unsuccessfully. Answers highlighted in red were considered incorrect. In the second example, both approaches are incorrect; the PDFTriage approach fetches the incorrect section, rather than just the first page, the chunk retrieval approach has no knowledge of document structure and paragraph order.

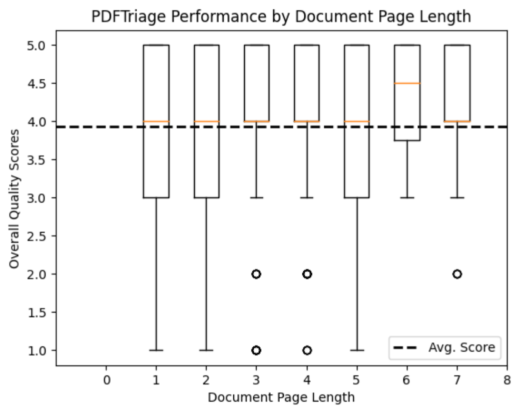


Figure 4: PDFTriage Performance compared to Document Page Length (uses "Overall Quality" scores)

6.4 PDFTriage performs consistently across document lengths

We also wanted to calculate the correlation between PDFTriage performance and the length of the document overall. Between the human-annotated PDFTriage answer score for "Overall Quality" and document length, we found a Pearson’s correlation coefficient of -0.015. This indicates that document length has a negligible effect on the efficacy of PDFTriage, strengthening the generalizability of our technique to both short and long documents.

The length of different document types seems to ultimately have no effect on overall performance. The ability of PDFTriage to query specific textual sections within the document prevents the need to ingest documents with excessively large contexts. It allows PDFTriage to connect disparate parts of a document for multi-page questions such as table reasoning, cross-page tasks, figure questions, and structure questions, prioritizing relevant context and minimizing irrelevant information. As a result, GPT-3 and other LLMs are better capable of handling the reduced context size and ultimately utilize less computational and financial resources for document QA tasks.

7 Future Work & Conclusions

In this work, we present PDFTriage, a novel question-answering technique specialized for document-oriented tasks. We compare our approach to existing techniques for question-answering, such as page retrieval and chunk retrieval, to demonstrate the strengths of our approach. We find that PDFTriage offers superior performance to existing approaches. PDFTriage also proves effective across various document lengths and contexts used for retrieval. We are considering the following directions for future work:

1. Developing multi-modal approaches that incorporate table and figure information into GPT-4 question-answering for documents.

2. Incorporate question type in PDFTriage approach to improve efficiency and efficacy of the approach.