arXiv:2402.05131v2 [cs.CL] 10 Feb 2024

Financial Report Chunking for Effective Retrieval Augmented Generation

Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li

Unstructured Technologies
Sacramento, CA, USA
`leah@unstructured.io`
`https://unstructured.io`

Abstract. Chunking information is a key step in Retrieval Augmented Generation (RAG). Current research primarily centers on paragraph-level chunking. This approach treats all texts as equal and neglects the information contained in the structure of documents. We propose an expanded approach to chunk documents by moving beyond mere paragraph-level chunking to chunk primary by structural element components of documents. Dissecting documents into these constituent elements creates a new way to chunk documents that yields the best chunk size without tuning. We introduce a novel framework that evaluates how chunking based on element types annotated by document understanding models contributes to the overall context and accuracy of the information retrieved. We also demonstrate how this approach impacts RAG assisted Question & Answer task performance. Our research includes a comprehensive analysis of various element types, their role in effective information retrieval, and the impact they have on the quality of RAG outputs. Findings support that element type based chunking largely improve RAG results on financial reporting. Through this research, we are also able to answer how to uncover highly accurate RAG.

Keywords: Retrieval Augmented Generation · Document Chunking · Document Pre-Processing · Financial Domain · Large Language Models

1 Introduction

Existing approaches for document understanding use a combination of methods from the computer vision and natural language processing domains to identify the different components in a document. In the rapidly evolving landscape of artificial intelligence, the capability to effectively process unstructured data is becoming increasingly critical. Large Language Models (LLMs) like GPT-4 have revolutionized natural language understanding and generation, as evidenced by their prompt-based functionalities [31], enabling a wide range of applications [5]. However, the efficacy of these models is often constrained by their reliance on the size and quality of the data they process. A notable limitation is the restricted contextual window of LLMs, which hampers their ability to fully comprehend the contents of extensive documents [25,22,18]. By dissecting large volumes of text into smaller, more focused segments, LLMs can process each part with greater precision, ensuring a thorough understanding of each section. This segmented approach allows for meticulous analysis of unstructured data, enabling LLMs to construct a more comprehensive and coherent understanding of the entire document [41]. There remains a challenge in ensuring factual accuracy and relevance in the generated responses, especially when dealing with complex or extensive information.

Recently, Retrieval Augmented Generation (RAG) [21,12] has been developed to address the *hallucination* problem with LLMs [15,43] when recovering factual information directly from an LLM. In RAG, instead of answering a user query directly using an LLM, the user query is used to retrieve documents or segments from a corpus and the top retrieved documents or segments are used to generate the answer in conjunction with an LLM. In this way, RAG constraints the answer to the set of retrieved documents. RAGs have been used as well to answer questions from single documents [14]. The documents are split into smaller parts or chunks, indexed by a retrieval system and recovered and processed depending on the user information need. In a sense, this process allows answering questions about information in a single document, thus contributing to the set of techniques available for document understanding.

Since documents need to be *chunked* for RAG processing, this raises the question about what is the best practice to chunk documents for effective RAG document understanding. There are several dimensions to consider when deciding how to chunk a document, which includes the size of the chunks.

The retrieval system in RAG can use traditional retrieval systems using bag-of-words methods or a vector database. If a vector database is used, then an embedding needs to be obtained from each chunk, thus the number of tokens in the chunk is relevant since the neural networks processing the chunks might have constraints on the number of tokens. As well, different chunk sizes might have undesirable retrieval results. Since the most relevant retrieved chunks need to be processed by an LLM, the number of tokens in retrieved chunks might have an effect in the generation of the answer [25]. As we see, chunking is required for RAG systems and there are several advantages and disadvantages when considering how to chunk a document.

In this work, we study specifically the chunking of U.S. Securities and Exchange Commission (SEC)[1] Financial Reports[2], including 10-Ks, 10-Qs, and 8-Ks. This study plays a critical role in offering insights into the financial health and operational dynamics of public companies. These documents present unique challenges in terms of document processing and information extraction as they consist of varying sizes and layouts, and contain a variety of tabular information. Previous work has evaluated the processing of these reports with simple chunking strategies (e.g., tokens), but we believe that a more effective use of these reports might be achieved by a better pre-processing of the documents

---

[1] `https://www.sec.gov`

[2] `https://www.sec.gov/files/cf-frm.pdf`

and chunking configuration[3] [14]. To the best of our knowledge, this is the first systematic study on chunking for document understanding and more specifically for processing financial reports.

2 Related work

RAG is an innovative method that has emerged to enhance the performance of LLMs by incorporating external knowledge, thereby boosting their capabilities. This technique has undergone substantial research, examining various configurations and applications. Key research includes Gao et al.'s [12] detailed analysis of RAG configurations and their role in enhancing Natural Language Processing (NLP) tasks, reducing errors, and improving factual accuracy. Several context retrieval methods are proposed to dynamically retrieve documents to improve the coherence of generated outputs [1]. Other research introduced advancements in RAG, including reasoning chain storage and optimization strategies for retrieval, respectively, broadening the scope and efficiency of RAG applications in LLMs [21]. More recent work has compared RAG vs LLM fine-tuning, and identified that applying both improves the performance of each individual method [2].

Chunking has been identified as the key factor in the success of RAG, improving the relevance of retrieved content by ensuring accurate embedding of text with minimal noise. Various strategies have been developed for text subdivision, each with its unique approach. They can be summarized as follows: the ***fixed size strategy*** divides text into uniform segments, but it often overlooks the underlying textual structure. In contrast, the ***recursive strategy*** iteratively subdivides text using separators like punctuation marks, allowing it to adapt more fluidly to the content. The ***contextual strategy*** takes this a step further by employing NLP techniques such as sentence segmentation to represent the meaning in context. Lastly, the ***hybrid strategy*** combines different approaches, offering greater flexibility in handling diverse text types [34]. However, an area yet to be explored in RAG chunking based on element types (document structure), which involves analyzing the inherent structure of documents, such as headings, paragraphs, tables, to guide the chunking process. Although chunking by Markdown and LaTeX comes closer to addressing element types, it's not the same in nature as a dedicated approach that directly considers document structure and element types for chunking, which could potentially yield more contextually relevant chunks.

Exploring the structure of financial reports is an exceptional area for establishing optimal principles for chunking. The intricate nature of document structures and contents has resulted in most of the work processing financial reports focusing on the identification of structural elements. Among previous work, we find El-Haj et al. [10] and the FinTOC challenges [17,4,11] that have worked at the document structure level for UK and French financial reports. Ad-

---

[3] `https://www.cnbc.com/2023/12/19/gpt-and-other-ai-models-cant-analyze-an-sec-filing-researchers-find.html`

ditionally, there is recent work that considers U.S. SEC reports, which includes DocLayNet [33] and more specifically with the report tables in FinTabNet [45].

On the side of financial models, there is work in sentiment analysis in finance [37], which includes the pre-training of specialised models such as FinBERT by Liu et al. [26], which is a BERT based model pre-trained on large corpora including large collections of financial news collected from different sites and FinBERT by DeSola et al, [9] trained on Wikipedia, BookCorpus and U.S. SEC data. Additional models include BloombergGPT [40], FinGPT [42] and Instruct-FinGPT[44].

More advance datasets in the financial domain include FinQA [6], LLMWare [27], ConFIRM [8] and TAT-QA [46] among others [7,38,19] that have been prepared for retrieval and or Questions and Answering (Q&A) tasks over snippets of financial data that includes tabular data, which has allowed methods on large language models to be tested on them [39].

Most of the previous work has focused on understanding the layout of financial documents or understanding specific snippets of existing reports with different levels of complexity, but there has not been much research in understanding financial report documents, except some more recent work that includes FinanceBench [14], in which a set of questions about the content of financial reports are proposed that includes the evidence snippet.

More specifically on document chunking methods for RAG, there are standard approaches being considered such as chunking text into spans of a given token length (e.g. 128 and 256) or chunking based on sentences. Open source projects already allow simple processing of documents (e.g. Unstructured[4], Llamaindex[5] or Langchain [6]), without explicitly considering the table structure on which these chunking strategies are applied.

Even though different approaches are available, an exhaustive evaluation of chunking applied to RAG and specifically to financial reporting, except for some limited chunking analysis [14,36], is non-existent. In our work, we compare a broad range of chunking approaches in addition to more simple ones and provide an analysis of the outcomes of different methods when asking questions about different aspects of the reports.

3 Methods

In this section, we present the chunking strategies that we have evaluated. Before describing the chunking strategies, we present the RAG environment in which these strategies have been evaluated and the dataset used for evaluation.

3.1 RAG setting for the experiments

The RAG pipeline used to process a user question is presented in figure 1 and is a common instance of a RAG [12]. Prior to answering any question about a given

---

[4] https://unstructured.io

[5] https://www.llamaindex.ai

[6] https://www.langchain.com

document, the document is split into chunks and the chunks are indexed into a vector database (vectordb). When a question is sent to the RAG system, the top-k chunks most similar to the question are retrieved from the vector database and used to generate the answer using a large language model as generator. In order to retrieve chunks from the vector database, the question is encoded into a vector that is compared to the vector previously generated from the chunks. To prompt the generator, the question is converted into a set of instructions that instruct the LLM to find the answer within the top-k retrieved chunks.

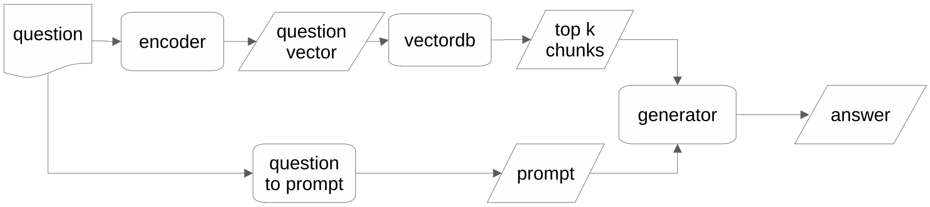

Fig. 1. RAG steps to answer a question about a document

In our experiments, we modify the way documents are *chunked* prior to being indexed in the vector database. All other settings remain constant. In the following sections, we describe in more detail each one of the components and processes used.

3.2 Indexing and retrieval

We have used the open source system Weaviate[7] as our vector database. As encoder model, we have used a sentence transformer [35] trained on over 256M questions and answers, which is available from the HuggingFace system[8].

As shown in figure 2, to index a document, first the document is split into chunks, then each chunk is processed by an encoder model and then indexed into the vector database. Based on the chunking strategy a document will be split into a larger or smaller set of chunks.

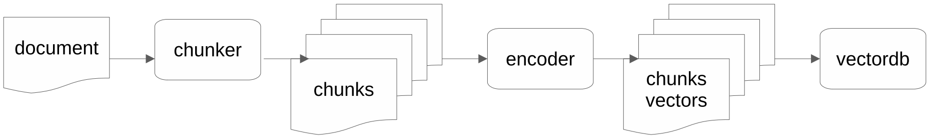

Fig. 2. Indexing of document chunks into the vector database

---

[7] https://weaviate.io/developers/weaviate

[8] https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1

As shown in figure 1, to retrieve chunks relevant to a question, the question is converted into a vector representation and the vector database returns a ranked list of chunks based on the similarity between question vector and the chunks in the database. Weaviate implements an approximate nearest neighbours algorithm [28] as their retrieval approach, which supports fast retrieval with high accuracy. In our experiments, we retrieve the top-10 chunks for each question.

3.3 Generation

Once the vector database has retrieved the top-10 chunks based on a question, the generation module generates the answer. To do so, a prompt based on the question and the retrieved chunks are provided to a large language model that generates the answer of the system.

We have used GPT-4 [31] as the generator, which has shown best performance compared to earlier versions. As well, its performance was better compared to existing open source alternatives [22] such as Mixtral [16]. We used the prompt presented in figure 3 that we designed on another similar RAG implementation with different document types. The prompt conditions the answer to the query and the chunks, referred to as *source*, and if the generator cannot answer it should return *No answer*.

```
please answer the question below by referencing the list of sources
provided after the question; if the question can not be answered just
respond 'No answer'. The sources are listed after "Sources:".

Question: {query}

Sources: {key} - {source}
...
```

Fig. 3. Example prompt template used by the generator

3.4 Chunking

As a baseline chunking method, we have split the documents into chunks of size n tokens ($n \in \{128, 256, 512\}$). As well, an aggregation of the output by the indexing of different chunking configurations has been considered.

In addition to chunking based on the number of tokens, we have processed the documents using computer vision and natural language processing to extract elements identified in the reports. The list of elements considered are provided by the Unstructured[9] open source library. From the set of processing strategies,

---

[9] `https://unstructured-io.github.io/unstructured/introduction.html#elements`

we use Chipper, a vision encoder decoder[10] model inspired by Donut [20] to showcase the performance difference. The Chipper model outputs results as a JSON representation of the document, listing elements per page characterized by their element type. Additionally, Chipper provides a bounding box enclosing each element on the page and the corresponding element text.

These elements are sometimes short to be considered as chunks, so to generate chunks from elements the following steps have been followed. Given the structure of finance reporting documents, our structural chunking efforts are concentrated on processing titles, texts, and tables. The steps to generate element-based chunks are:

- if the element text length is smaller than 2,048 characters, a merge with the following element is attempted
- iteratively, element texts are merged following the step above till either the desired length is achieved, without breaking the element
- if a title element is found, a new chunk is started
- if a table element is found, a new chunk is started, preserving the entire table

After element-based chunks have been derived, three types of metadata are generated to enrich the content and support efficient indexing. The first two types, generated via predefined prompt templates with GPT-4, include: 1) up to 6 representative keywords of the composite chunk 2) a summarised paragraph of the composite chunk. The third type is 3) Naive representation using the first two sentences from a composite chunk (a kind of prefix) and in the case of tables, the description of the table, which is typically identified in the table caption.

3.5 Dataset

To evaluate the performance of the different chunking approaches, we have used the FinanceBench dataset [14]. FinanceBench is a new benchmarking dataset designed to assess the capabilities of LLMs in answering open-book financial questions. The questions collected are realistic and applicable to real-world financial scenarios and include complex questions that require computational reasoning to arrive at conclusive answers.

This dataset is made of 150 instances with questions and answers from 84 unique reports. The dataset does not include the source documents, which we have downloaded. We were able to recover only 80 documents, which reduces the number of questions to 141 from the original 150. The distribution of Unstructured elements predictions are shown in table 1.

Documents have a varying number of pages, spanning from 4 pages (FOOTLOCKER_2022_8K_dated-2022-05-20) to 549 pages (e.g. PEPSICO_2021_10K), with an average of 147.34 with std 97.78 with a total of 11,787 pages combined. Each instance contains a link to the report, the question, a question type , the answer and supporting evidence, with page number where the evidence is located

---

[10] `https://huggingface.co/docs/transformers/model_doc/vision-encoder-decoder`

Table 1. Unstructured element types distribution for *Chipper* predictions against documents in FinanceBench.

Element Type	***Chipper* Entities**
NarrativeText	61,780
Title	29,664
ListItem	33,054
UncategorizedText	9,400
Footer	1,026
Table	7,700
Header	3,959
Image	26
FigureCaption	54
Formula	29
Address	229
Total	**146,921**

in the document, that allows for a closer evaluation of the results. Based on the page number, evidence contexts are located in different areas in the documents, ranging from the first page in some cases up to page 304 in one instance. The mean page number to find the evidence is 54.58 with a standard deviation of 43.66, which shows that evidence contexts to answer the questions are spread within a document. These characteristics make FinanceBench a perfect dataset for evaluating RAG. An example instance is available in table 2.

4 Results

In this section, we evaluate the different chunking strategies using the FinanceBench dataset. Our evaluation is grounded in factual accuracy, which allows us to measure the effectiveness of each configuration by its precision in retrieving answers that match the ground truth, as well as its generation abilities.

We are considering 80 documents and 141 questions from FinanceBench. Using the OpenAI tokenizer from the model *text-embedding-ada-002* that uses the tokenizer *cl100k_base*[11], there are on average 102,444.35 tokens with std of 61,979.45, which shows the large variability of document lengths as seen by the different number of pages per document presented above.

Chunking Efficiency The first thing we analyzed is the total number of chunks, as it impacts indexing time. We would like to observe the relationship between accuracy and total chunk size. Table 3 shows the number of chunks derived from each one of the processing methods. Unstructured element-based chunks are closer in size to Base 512, and as the chunk size decreases for the basic chunking strategies, the total number of chunks increases linearly.

---

[11] `https://platform.openai.com/docs/guides/embeddings/limitations-risks`

Table 2. Example question from the FinanceBench dataset

Field	Value
financebench_id	financebench_id_00859
doc_name	VERIZON_2021_10K
doc_link	https://www.verizon.com/about/sites/default/files/2021-Annual-Report-on-Form-10-K.pdf
question_type	'novel-generated'
question	Among all of the derivative instruments that Verizon used to manage the exposure to fluctuations of foreign currencies exchange rates or interest rates, which one had the highest notional value in FY 2021?
answer	Cross currency swaps. Its notional value was $32,502 million.,
evidence_text	Derivative Instruments We enter into derivative transactions primarily to manage our exposure to fluctuations in foreign currency exchange rates and interest rates. We employ risk management strategies, which may include the use of a variety of derivatives including interest rate swaps, cross currency swaps, forward starting interest rate swaps, treasury rate locks, interest rate caps, swaptions and foreign exchange forwards. We do not hold derivatives for trading purposes. The following table sets forth the notional amounts of our outstanding derivative instruments: (dollars in millions) At December 31, 2021 2020 Interest rate swaps $ 19,779 $ 17,768 Cross currency swaps 32,502 26,288 Forward starting interest rate swaps 1,000 2,000 Foreign exchange forwards 932 1,405
page_number	85

Table 3. Chunks statistics for basic chunking elements and Unstructured elements

Processing	total chunks	mean chunks per document (std)	tables mean (std)
Base 128	64,058	800.73 (484.11)	N/A
Base 256	32,051	400.64 (242.04)	N/A
Base 512	16,046	200.58 (121.01)	N/A
Chipper	20,843	260.57 (145.80)	96.20 (57.53)

Retrieval Accuracy Secondly, we evaluate the capabilities of each chunking strategy in terms of retrieval accuracy. We use the page numbers in the ground truth to calculate the page-level retrieval accuracy, and we use ROUGE [24] and BLEU [32] scores to evaluate the accuracy of paragraph-level retrieval compared to the ground truth evidence paragraphs.

As shown in Table 4, when compared to Unstructured element-based chunking strategies, basic chunking strategies seem to have higher page-level retrieval accuracy but lower paragraph-level accuracy on average. Additionally, basic chunking strategies also lack consistency between page-level and paragraph-level accuracy; higher page-level accuracy doesn't ensure higher paragraph-level accuracy. For example, Base 128 has the second highest page-level accuracy but the lowest paragraph-level scores among all. On the other hand, element-based chunking strategies showed more consistent results.

A fascinating discovery is that when various chunking strategies are combined, it results in enhanced retrieval scores, achieving superior performance at both the page level (84.4%) and paragraph level (with ROUGE at 0.568% and BLEU at 0.452%). This finding addresses an unresolved question: how to improve the accuracy of RAG.

The element based method provides the highest scores and it also provides a mechanism to chunk documents without the need to fine tune hyper-parameters like the number of tokens in a chunk. This suggests the element based method is more generalizable and can be applied to new types of documents.

Q&A Accuracy Third, we evaluate the Q&A accuracy for the chunking strategies. In addition to manual evaluation, we have investigated an automatic evaluation using GPT-4. GPT-4 compares how the answers provided by our method are similar to or different from the FinanceBench gold standard, similar approaches have been previously evaluated [13,23,29,30]. The automatic evaluation allows scaling the evaluation efforts for the different chunking strategies that we have considered. We used the prompt template in figure 4.

```
Begin with True or False. Are the two following answers (Answer 1 and
Answer 2) the same with respect to the question between single quotes
'{question}'?

Answer 1: '{ground_truth_answer}'
Answer 2: '{generated_answer}'
```

Fig. 4. Evaluation prompt template. The {*question*}, {*ground_truth_answer*} and {*generated_answer*} fields are substituted for each question accordingly.

Results in table 5 show that element-based chunking strategies offer the best question-answering accuracy, which is consistent with page retrieval and paragraph retrieval accuracy.

Lastly, our approach stands out for its efficiency. Not only is element-based chunking generalizable without the need to select the chunk size, but when compared to the aggregation results that yield the highest retrieval scores. Element-based chunking achieves the highest retrieval scores with only half the number of chunks required compared to methods that do not consider the structure of the documents (62,529 v.s. 112,155). This can reduce the indexing cost and improve query latency because there are only half as many vectors to index for the vectordb that stores the chunks. This underscores the effectiveness of our solution in optimizing the balance between performance and computational resource requirements.

Table 4. Retrieval results. For each chunking strategy, we show the number of chunks for all the documents (Total Chunks), Page Accuracy, and ROUGE and BLEU scores. ROUGE and BLEU are calculated as the maximum score from the list of recovered contexts for a question when compared to the known evidence for that question.

Chunking strategy	Total Chunks	Page Accuracy	ROUGE	BLEU
Base 128	64,058	72.34	0.383	0.181
Base 256	32,051	73.05	0.433	0.231
Base 512	16,046	68.09	0.455	0.250
Base Aggregation	112,155	83.69	0.536	0.277
Keywords Chipper	20,843	46.10	0.444	0.315
Summary Chipper	20,843	62.41	0.473	0.350
Prefix & Table Description Chipper	20,843	67.38	0.514	0.400
Chipper Aggregation	62,529	**84.40**	**0.568**	**0.452**

Table 5. Q&A results. We show the percentage of questions with no answer and as well the accuracy either estimated automatically using GPT-4 or manually.

Chunking strategy	No answer	GPT-4	Manual
Base 128	35.46	29.08	35.46
Base 256	25.53	32.62	36.88
Base 512	24.82	41.84	48.23
Keywords Chipper	22.70	**43.97**	**53.19**
Summary Chipper	17.73	**43.97**	51.77
Prefix & Table Description Chipper	20.57	41.13	**53.19**

5 Discussion

Results demonstrate the efficacy of our approach in utilizing structural elements for chunking, which has enabled us to attain state-of-the-art performance on Q&A tasks within the FinanceBench dataset (accuracy of 50% vs 53.19%) when an index is created from document chunks and used for generation. This method, which we refer to as *element base chunking*, has shown to yield consistent results between retrieval and Q&A accuracy.

We have observed that using basic 512 chunking strategies produces results most similar to the Unstructured element-based approach, which may be due to the fact that 512 tokens share a similar length with the token size within our element-based chunks and capture a long context, but fail keep a coherent context in some cases, leaving out relevant information required for Q&A. This is further observed when considering the ROUGE and BLEU scores in table 4, where the chunk contexts for the baseline have lower scores.

These findings support existing research stating that the best basic chunk size varies from data to data [3]. These results show, as well, that our method adapts to different documents without tuning. Our method relies on the structural information that is present in the document's layout to adjust the chunk size automatically.

We have evaluated aggregating the output of different chunking methods in the retrieval experiments as sown in table 4. Even though the aggregation seems to be effective for retrieval, the Q&A exceeded the GPT-4 token limit, which resulted in a non-effective Q&A solution using the selected model.

As well, we evaluated variations of the prompt used to generate the answers (see figure 3). Re-ordering the retrieval context and the question, but results were not statistically different. We experimented as well with variations of the verbs using in the prompt, e.g. changing *referencing* with *using*, which seemed to lower the quality of the answers generated. This shows that prompt engineering is a relevant factor in RAG.

We evaluated using GPT-4 for evaluation instead of relying on manual evaluation. In most cases, GPT-4 evaluated correctly but failed when a more elaborate answer is provided. As shown in figure 5, the answer is 39.7% while the estimated answer is 39.73% but with a detailed explanation of the calculation.

```
Question: 'What is Coca Cola's FY2021 COGS % margin? Calculate what
was asked by utilizing the line items clearly shown in the income
statement.'?

Answer 1: '39.7%'
Answer 2: 'From the income statement referenced on page 60 of
COCACOLA_2021_10K_embedded.json, we can see that Coca Cola's total
revenue in FY2021 was $38,655 million and their cost of goods sold
(COGS) was $15,357 million. To calculate the COGS % margin, we divide
the COGS by the total revenue and multiply by 100:
(15,357 / 38,655) * 100 = 39.73%
So, Coca Cola's FY2021 COGS % margin was approximately 39.73%.'
```

Fig. 5. Evaluation prompt template

6 Conclusions and Future Work

Results show that our element based chunking strategy improves the state-of-the-art Q&A for the task, which is achieved by providing a better chunking strategy for the processed documents. We provide comparison with baseline chunking strategies that allow us to draw conclusions about different chunking methods.

As future work, we would like to perform a similar evaluation in other domains, e.g. biomedical, to understand how our findings apply outside financial reporting. As well, we would like studying which additional element types and/or relation between elements would support better chunking strategies for RAG.

Furthermore, we would like to study the impact of RAG configuration and element type based chunking.

References

1. Anantha, R., Bethi, T., Vodianik, D., Chappidi, S.: Context Tuning for Retrieval Augmented Generation (2023)
2. Balaguer, A., Benara, V., de Freitas Cunha, R.L., de M. Estevão Filho, R., Hendry, T., Holstein, D., Marsman, J., Mecklenburg, N., Malvar, S., Nunes, L.O., Padilha, R., Sharp, M., Silva, B., Sharma, S., Aski, V., Chandra, R.: Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture (2024)
3. Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., Abdelrazek, M.: Seven Failure Points When Engineering a Retrieval Augmented Generation System (2024)
4. Bentabet, N.I., Juge, R., El Maarouf, I., Mouilleron, V., Valsamou-Stanislawski, D., El-Haj, M.: The financial document structure extraction shared task (fintoc 2020). In: Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation. pp. 13–22 (2020)
5. Chen, H., Jiao, F., Li, X., Qin, C., Ravaut, M., Zhao, R., Xiong, C., Joty, S.: ChatGPT's One-year Anniversary: Are Open-Source Large Language Models Catching up? arXiv preprint arXiv:2311.16989 (2023)
6. Chen, Z., Chen, W., Smiley, C., Shah, S., Borova, I., Langdon, D., Moussa, R., Beane, M., Huang, T.H., Routledge, B., et al.: Finqa: A dataset of numerical reasoning over financial data. arXiv preprint arXiv:2109.00122 (2021)
7. Chen, Z., Li, S., Smiley, C., Ma, Z., Shah, S., Wang, W.Y.: ConvFinQA: Exploring the Chain of Numerical Reasoning in Conversational Finance Question Answering (2022)
8. Choi, S., Gazeley, W., Wong, S.H., Li, T.: Conversational Financial Information Retrieval Model (ConFIRM). arXiv preprint arXiv:2310.13001 (2023)
9. DeSola, V., Hanna, K., Nonis, P.: Finbert: pre-trained model on sec filings for financial natural language tasks. University of California (2019)
10. El-Haj, M., Rayson, P., Young, S., Walker, M.: Detecting document structure in a very large corpus of UK financial reports. European Language Resources Association (ELRA) (2014)
11. El Maarouf, I., Kang, J., Azzi, A.A., Bellato, S., Gan, M., El-Haj, M.: The financial document structure extraction shared task (FinTOC2021). In: Proceedings of the 3rd Financial Narrative Processing Workshop. pp. 111–119 (2021)
12. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H.: Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997 (2023)
13. Hada, R., Gumma, V., de Wynter, A., Diddee, H., Ahmed, M., Choudhury, M., Bali, K., Sitaram, S.: Are large language model-based evaluators the solution to scaling up multilingual evaluation? arXiv preprint arXiv:2309.07462 (2023)
14. Islam, P., Kannappan, A., Kiela, D., Qian, R., Scherrer, N., Vidgen, B.: FinanceBench: A New Benchmark for Financial Question Answering. arXiv preprint arXiv:2311.11944 (2023)
15. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of Hallucination in Natural Language Generation. ACM Computing Surveys **55**(12), 1–38 (Mar 2023). https://doi.org/10.1145/3571730, `http://dx.doi.org/10.1145/3571730`

16. Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., de las Casas, D., Hanna, E.B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L.R., Saulnier, L., Lachaux, M.A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T.L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E.: Mixtral of Experts (2024)
17. Juge, R., Bentabet, I., Ferradans, S.: The fintoc-2019 shared task: Financial document structure extraction. In: Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019). pp. 51–57 (2019)
18. Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., McHardy, R.: Challenges and applications of large language models. arXiv preprint arXiv:2307.10169 (2023)
19. Kaur, S., Smiley, C., Gupta, A., Sain, J., Wang, D., Siddagangappa, S., Aguda, T., Shah, S.: REFinD: Relation Extraction Financial Dataset. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '23, ACM (Jul 2023). https://doi.org/10.1145/3539618.3591911, `http://dx.doi.org/10.1145/3539618.3591911`
20. Kim, G., Hong, T., Yim, M., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., Park, S.: Donut: Document understanding transformer without ocr. arXiv preprint arXiv:2111.15664 **7**, 15 (2021)
21. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
22. Li, D., Shao, R., Xie, A., Sheng, Y., Zheng, L., Gonzalez, J.E., Stoica, I., Ma, X., Zhang, H.: How Long Can Open-Source LLMs Truly Promise on Context Length? (June 2023), `https://lmsys.org/blog/2023-06-29-longchat`
23. Li, Y., Duan, Y.: The evaluation of experiments of artificial general intelligence with gpt-4 based on dikwp. arXiv preprint (2023)
24. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out. pp. 74–81 (2004)
25. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts. arXiv preprint arXiv:2307.03172 (2023)
26. Liu, Z., Huang, D., Huang, K., Li, Z., Zhao, J.: Finbert: A pre-trained financial language representation model for financial text mining. In: Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence. pp. 4513–4519 (2021)
27. llmware: Rag Instruct Benchmark Tester. `https://huggingface.co/datasets/llmware/rag_instruct_benchmark_tester`, Accessed: January 15, 2024
28. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence **42**(4), 824–836 (2018)
29. Moore, S., Nguyen, H.A., Chen, T., Stamper, J.: Assessing the quality of multiple-choice questions using gpt-4 and rule-based methods. In: European Conference on Technology Enhanced Learning. pp. 229–245. Springer (2023)
30. Naismith, B., Mulcaire, P., Burstein, J.: Automated evaluation of written discourse coherence using gpt-4. In: Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023). pp. 394–403 (2023)
31. OpenAI, :, Achiam, J., Adler, S., Agarwal, S., et al.: GPT-4 Technical Report (2023)

32. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics. pp. 311–318 (2002)
33. Pfitzmann, B., Auer, C., Dolfi, M., Nassar, A.S., Staar, P.: Doclaynet: A large human-annotated dataset for document-layout segmentation. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 3743–3751 (2022)
34. Pinecone: Chunking strategies for llm applications, `https://www.pinecone.io/learn/chunking-strategies/`
35. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (11 2019), `https://arxiv.org/abs/1908.10084`
36. Retteter, J.: Mastering Table Extraction: Revolutionize Your Earnings Reports Analysis with AI. `https://medium.com/unstructured-io/mastering-table-extraction-revolutionize-your-earnings-reports-analysis-with-ai-1bc32c22720e`, Accessed: January 15, 2024
37. Rizinski, M., Peshov, H., Mishev, K., Jovanovik, M., Trajanov, D.: Sentiment Analysis in Finance: From Transformers Back to eXplainable Lexicons (XLex) (2023)
38. Shah, R.S., Chawla, K., Eidnani, D., Shah, A., Du, W., Chava, S., Raman, N., Smiley, C., Chen, J., Yang, D.: WHEN FLUE MEETS FLANG: Benchmarks and Large Pre-trained Language Model for Financial Domain (2022)
39. Singh Phogat, K., Harsha, C., Dasaratha, S., Ramakrishna, S., Akhil Puranam, S.: Zero-Shot Question Answering over Financial Documents using Large Language Models. arXiv e-prints pp. arXiv–2311 (2023)
40. Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., Mann, G.: BloombergGPT: A Large Language Model for Finance (2023)
41. Xu, P., Ping, W., Wu, X., McAfee, L., Zhu, C., Liu, Z., Subramanian, S., Bakhturina, E., Shoeybi, M., Catanzaro, B.: Retrieval meets Long Context Large Language Models (2023)
42. Yang, H., Liu, X.Y., Wang, C.D.: FinGPT: Open-Source Financial Large Language Models (2023)
43. Ye, H., Liu, T., Zhang, A., Hua, W., Jia, W.: Cognitive Mirage: A Review of Hallucinations in Large Language Models (2023)
44. Zhang, B., Yang, H., Liu, X.Y.: Instruct-FinGPT: Financial Sentiment Analysis by Instruction Tuning of General-Purpose Large Language Models (2023)
45. Zheng, X., Burdick, D., Popa, L., Zhong, X., Wang, N.X.R.: Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 697–706 (2021)
46. Zhu, F., Lei, W., Huang, Y., Wang, C., Zhang, S., Lv, J., Feng, F., Chua, T.S.: TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. arXiv preprint arXiv:2105.07624 (2021)