CarbonPDF: Question-Answering Reasoning Framework for Assessing Product Carbon Footprint in PDF Documents

Anonymous ACL submission

Abstract

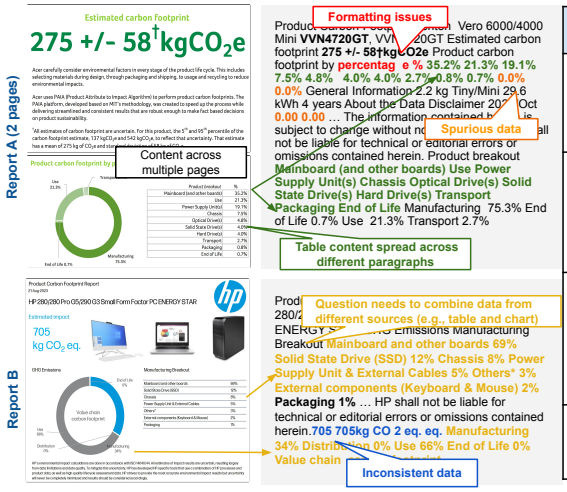
Product sustainability reports provide valuable insights into the environmental impacts of a product and are often distributed in PDF format. These reports often include a combination of tables and text, which complicates their analysis. The lack of standardization and the variability in reporting formats further exacerbate the difficulty of extracting and interpreting relevant information from large volumes of documents. In this paper, we tackle the challenge of answering questions related to carbon footprints within sustainability reports available in PDF format. Unlike previous approaches, our focus is on addressing the difficulties posed by the unstructured and inconsistent nature of text extracted from PDF parsing. To facilitate this analysis, we introduce CarbonPDF-QA, an open-source dataset containing question-answering pairs for each document, along with human-annotated answers. Our evaluation of GPT-4 on this dataset reveals its inadequacy in answering questions based on inconsistent data. To address this limitation, we propose CarbonPDF, an LLM-based technique specifically designed to answer carbon footprint questions on such datasets. We develop CarbonPDF by fine-tuning Llama 3 with our training data. Our results show that our technique outperforms current state-of-the-art techniques, including question-answering (QA) systems finetuned on table and text data.

1 Introduction

As the climate crisis becomes more urgent, sustainability reporting has become increasingly important, compelling companies and organizations to disclose their environmental impacts and sustainability efforts (Olivier M. Schwab, 2022; Rodriguez, Isabel and Caglio, Ariela, 2023). This reporting is essential not only for regulatory compliance but also for demonstrating corporate responsibility and transparency to stakeholders. To conduct thorough analyses, stakeholders like regulators and consumers rely on these reports to engage in carbon footprint assessments and compliance checks. These analyses help evaluate the environmental impact of products and ensure that companies adhere to sustainability commitments and standards. However, the lack of standardization and the complex format of these reports containing hybrid data — a mix of tables and text — presents significant challenges for effective analysis. Disparities in how data is presented make it difficult to perform numerical reasoning and make it challenging to compare and assess sustainability metrics across different companies or even within the same organization over time in an automated manner.

Recent studies have investigated the use of question-answering (QA) techniques for analyzing numerical information in hybrid data by framing data-related analysis as questions (Zhu et al., 2021; Chen et al., 2022). These approaches use language models to interpret the hybrid data and perform numerical reasoning to streamline the analysis process. A common approach involves feeding the hybrid data and specific questions into language models to generate answers (Zhu et al., 2024).

However, analyzing hybrid data in carbon sustainability reports presents significant challenges. These difficulties arise because reports are often available as Portable Document Format (PDF) documents, and extracting hybrid data from PDFs is often error-prone. For instance, although tables may appear structured, PDF does not encode this information as tables, unlike HTML or spreadsheets. Instead, PDFs represent tables as a collection of text and lines placed at specific coordinates without any explicit information about rows or columns. This lack of inherent structure makes it difficult to extract and reconstruct tables accurately, as extraction algorithms must infer relationships between text elements based on their positions, which can be complex and unreliable. As shown in Figure 1, the data extracted from a PDF may appear in a



Reasoning	Question	Answer	Derivation
Word Matching	What are the carbon footprint percentages of the power supply unit and packaging in the VVN4720GT laptop?	[19.1%,0.8%]	N/A
Max/Min	What is the component with the lowest carbon footprint percentage in the manufacturing breakdown of the VVN4720GT desktop?	{"Packaging":0.7%}	Collect all percentages; Call min( );
Top 3/5	What are the top 3 components with the highest carbon footprint percentages in the manufacturing breakdown of the VVN4720GT laptop?	[{"Mainboard":35.2 %}, {"Power":19.1%}, {"Chassis":7.5%}]	Collect all percentages; Sort; Show top 3;
Calculation	What is the carbon footprint of chassis in the VVN4720GT laptop?	20.625	275×7.5%

Figure 1: CarbonPDF-QA dataset is collected from product carbon reports. Examples highlight the challenges of extracting table content from PDF documents. The table provides an overview of the different question types over the unstructured tabular and text data in the dataset.

different order than expected, even if it looks sequential in the document. This happens because the visual layout of the table does not always match a clear, structured format within the PDF file.

The problem is further complicated by variations in how different documents represent tables internally. Content may be spread across different pages or sections, making connections between related data loose or unclear. Additionally, hidden text and numbers encoded within the PDF may not be visible but can be read using programs, resulting in spurious or inconsistent data. Existing state-of-the-art QA systems that handle hybrid data generally assume a structured table format, where the content is free from such anomalies (Zhu et al., 2024). Thus, these systems may struggle when presented with inconsistent content extracted from PDF documents, where table and text data are represented for visual presentation rather than data analysis. Moreover, most QA systems are typically designed to handle reasoning questions over a single table. This limits their effectiveness when dealing with content that spans multiple tables.

In this paper, we address the challenges associated with the problem of hybrid data extracted from sustainability report PDF documents. Our goal is to answer carbon footprint-related questions based on this extracted data from PDF, tackling the challenges posed by inconsistent and loosely connected table and text content. We refer to this data as inconsistent because we do not modify it to remove spurious information. Additionally, numbers and text often misalign and can be scattered across multiple paragraphs, complicating direct question answering. To facilitate analysis, we present the CarbonPDF dataset — an open-domain carbon product report in PDF format. This dataset includes a variety of carbon assessment numerical reasoning questions that require extracting information from text, tables, and graphical charts. We developed this dataset through a combination of automated processes and human verification to ensure its accuracy and reliability. To the best of our knowledge, CarbonPDF-QA is the first dataset specifically designed to include inconsistent data, addressing the unique challenges of analyzing unstructured tabular and text content.

We also explore how LLMs can effectively manage complex and inconsistent data from sources such as PDFs. We propose CarbonPDF, built on Llama 3, which not only extracts evidence from retrieved documents but also generates executable workflows to address a variety of questions. This approach improves the system's reasoning capabilities, allowing it to focus on relevant evidence and deliver more accurate answers. In summary, we make the following contributions.

- We introduce CarbonPDF-QA dataset, an open-domain question-answering benchmark for carbon product PDF documents that contain unstructured table and text data. The dataset was created using reports from different companies, with ground truth answers that were manually verified by humans.

- We develop the CarbonPDF model, a QA sys-

Table 1: Statistics of the CarbonPDF-QA dataset

PDF Statistic		QA Dataset Summary		
Type		**Ques. Type**	**Train**	**Test**
# Company	4	Word Match	8681	1959
# File	1737	Max/Min	1934	487
Avg.char./file	3772	Top 3/5	1245	369
Avg.words/file	563	Calculation	7648	2062
Avg.pages/file	1.74	Total Ques.	19508	4877

tem designed to handle the complexities of inconsistent or spurious data extracted from PDF documents.

- We conduct extensive experiments and demonstrate that our model outperforms existing state-of-the-art techniques, including RAG and QA systems. Additionally, we perform detailed analyses to showcase the model's capabilities in handling complex numerical reasoning on unstructured table and text data.

2 CarbonPDF-QA Dataset

2.1 Data Collection

Our datasets are derived from computing products' carbon footprint reports, as shown in the left part of Table 1. We collected 1,737 PDF reports from the websites of HP (HP Inc., 2024), Dell (Dell Inc., 2024), Acer (Acer Inc., 2024), and Lenovo (Lenovo Inc., 2024). Each file contains, on average, around 4,000 characters and 2 pages. To process these reports, we utilized the PyMuPDF library (PyMuPDF Developers, 2024) to open, parse, and convert the PDF files into text. We developed custom parsers to extract both product specifications, such as product name, display size, and product weight, as well as carbon-related information, including the total product carbon footprint (PCF) and the carbon footprint percentage of each component in the manufacturing carbon footprint breakdown. The extract text and values are stored in CSV files.

2.2 Dataset Preparation

Question Generation The dataset includes various question types, shown in the right part of Table 1. These range from word-matching questions, where answers can be directly extracted from the PDF file, such as the total product carbon footprint or the carbon footprint percentage of a specific component, to more complex questions. The latter requires not only evidence extraction from the PDF document that spans different sections but also arithmetic calculations to derive the final answers. The evidence extraction annotations help to determine whether the model correctly identifies and uses the necessary information to answer the question accurately. The questions in the dataset focus on various aspects of product carbon footprints, including those related to individual components or multiple components of a product. For each product document, we generate, on average, at least 14 questions that can be answered using information from the PDFs. As a result, each question is paired with at least one reference document that provides the necessary context to answer the question.

Reference Relevance The CarbonPDF-QA dataset samples also include reference text that may not be relevant to answering the questions. These scenarios are included to reflect real-world challenges where not all documents are pertinent, requiring the model to focus on the relevant data to answer the questions accurately. To do so, the dataset includes a relevance token that indicates whether the reference text is pertinent to the given question. When generating the dataset, we set the relevance token to True for reference texts that can be used to answer the questions. To create irrelevant references, we developed a program that selectively removes key information necessary for answering the questions. For example, some product component breakdowns might be removed. These modified texts are then annotated as irrelevant. The irrelevant samples make up approximately 30% of the entire dataset.

Evidence Annotation For each question, the dataset also includes the location of the information within the reference text needed to answer the questions. To achieve this, we treat the entire reference text as a character array and use the array index to pinpoint the relevant evidence. This approach helps mitigate spurious or inconsistent text issues when extracting content from PDFs. For example, the PDF parser may extract inconsistent data, such as "705 705kg CO2 eq. eq." instead of "705kg CO2 eq." (see Figure 1). Thus, evidence location can be used to extract the relevant information and ignore any spurious text. We represent the evidence information as follows. For each question-text pair, we create a JSON object to store the evidence. The keys in this object correspond to the start and end indices of the evidence within the character array, and the value is the evidence text.

Program Generation For all the questions, the dataset includes a program script to generate the

output. To achieve this, we manually write a Python script for each question-text pair, using simple arithmetic expressions to compute the final answer. These scripts are designed to be straightforward and do not require any external library imports. Our dataset also includes questions that may yield multiple answers, such as queries about the carbon footprint of both an HDD and a chassis. In such cases, the final answer is structured as a list, with the order of components corresponding to the sequence specified in the question. Finally, the entire dataset is split into a training set and a test set with an 80/20 ratio.

Data Validation To validate the data quality, we enlisted five students to verify the ground truth data. These students conducted the data validation as part of their class projects, which also involved developing tools for analyzing carbon reports. We divided the students into two teams, distributing the PDFs equally, and extracted content among them for validation. The verification process was a combination of automated checks and manual review. First, we used programs to ensure that extracted values fell within reasonable ranges — such as percentages not exceeding 100% or carbon footprint values not being excessively high. Each team, consisting of at least two students, was tasked with verifying each extracted value. To further ensure accuracy, we plotted the data to visually identify potential errors, such as outliers. If outliers were detected, we revisited the original PDF files and manually verified the data. We also checked that the evidence aligned with the questions being asked. To verify the accuracy of the ground truth indices, we printed out the extracted text alongside their corresponding indices from the document and conducted a visual inspection for any errors. Finally, we executed all the generated programs and compared their results with the ground truth to verify the accuracy of the programs. If the results matched, the program is deemed correct.

3 CarbonPDF Design

3.1 Overview

A key design goal of CarbonPDF is to provide accurate, fact-based answers to user queries. However, previous research shows that state-of-the-art LLMs often struggle with maintaining factual accuracy (Mallen et al., 2022). To mitigate this issue, we incorporate Retrieval Augmented Generation (RAG) techniques into our design strategy, leveraging their success in reducing factual errors in knowledge-intensive tasks. Note that unlike prior reasoning methods, such as TAT-LLM (Zhu et al., 2024), which assume that the correct context is always provided, our approach recognizes that real-world scenarios often involve ambiguous or even misleading context — a challenge we address in this work.

Figure 2 illustrates our approach's key components and overall workflow. For a given question, CarbonPDF first retrieves relevant context from the PDF database. The retriever finds the most relevant PDF document that might contain the answer. The retrieved reference text, which includes unstructured PDF data, is then combined with the question and a set of instructions to guide CarbonPDF in its reasoning process to derive the final answer. For additional details on the instruction prompt template, please refer to the Appendix A.1.

The reasoning process involves several key steps to derive the final answer. Initially, CarbonPDF assesses the relevance of the retrieved reference to the question by generating a relevance token, which helps determine whether the content provided can effectively answer the question. The relevance token is binary, with a True or False value. If the reference text is irrelevant, CarbonPDF returns with no answer. Although the system can be configured to retry by retrieving a different reference, we do not exhaustively search through all possible references. This provides a lower bound on our system's performance.

Conversely, if the reference text is found to be relevant, CarbonPDF proceeds with evidence extraction. In this step, CarbonPDF identifies the specific portions of the reference text necessary to answer the question. The evidence includes text and array indices that pinpoint the locations of relevant text within the reference's character array. With this extracted evidence, CarbonPDF generates a program to produce the final answer. A program interpreter then executes the necessary calculations and generates the final response.

3.2 Retriever

Document retrieval has traditionally identified relevant documents through keyword matching. Recently, neural network-based approaches, such as Contriever (Izacard et al., 2021), which utilize neural embeddings for retrieval, have been introduced and employed in models like Self-RAG (Asai et al., 2023). However, in our work, we chose to use Term

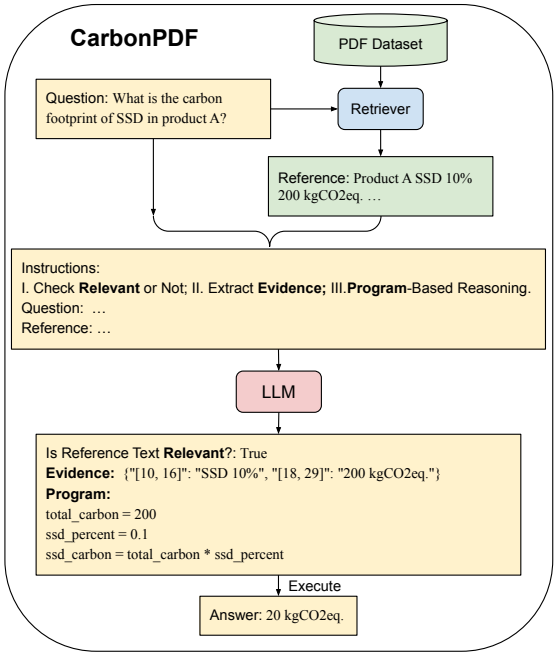


Figure 2: Our main design

Frequency-Inverse Document Frequency (TF-IDF) embedding because we found them to be both efficient and effective for our use case. Although we currently use TF-IDF, our approach is flexible and can integrate other retrieval techniques, including models like Contriever.

To retrieve the relevant documents, we first convert the entire document corpus into TF-IDF embeddings using the *sklearn.TfidfVectorizer* function. This function transforms each document into a vector of numerical values, where each dimension represents a term in the corpus, and the value in each dimension corresponds to the term's TF-IDF score. When CarbonPDF receives a question, it also converts this question into a TF-IDF vector. We then compute the cosine similarity between the question vector and the TF-IDF vectors of the documents in our corpus. This similarity metric produces a ranking of documents based on their relevance to the question. The system then selects the document with the highest similarity score as the closest match to the query.

3.3 Program-based Reasoning

LLMs often struggle with reasoning questions that involve complex calculations (Lewkowycz et al., 2022). Recently, program-aided language models have demonstrated effectiveness in overcoming these challenges by leveraging programmatic reasoning (Gao et al., 2023). This approach improves the model's ability to perform complex calculations and produce accurate answers. Building on this insight, we finetune CarbonPDF LLM to generate a Python program to compute the results based on the extracted evidence.

The final program generated by CarbonPDF varies based on the type of question and the document's content. Some carbon documents provide the total carbon footprint along with lifecycle breakdowns (e.g., manufacturing, end-of-life, and transport) and detailed breakdowns for individual components (e.g., HDDs, chassis). Other documents may report the carbon footprint of individual components directly without offering a comprehensive lifecycle breakdown.

Similarly, the complexity of the questions may also affect the program generated. For instance, questions requiring detailed calculations for individual components, which depend on factors like the manufacturing process, involve more complex reasoning. To handle such complexity, CarbonPDF employs a multistep approach in its generated programs. Unlike single-step calculations, we store intermediate values in variables and then perform necessary multiplications to derive the answer. In situations with multiple answers, CarbonPDF produces the final answers as a list, maintaining the specified question order.

3.4 Training

We train our CarbonPDF model by fine-tuning Llama 3_{8B} (Meta AI, 2023) using two NVIDIA RTX 6000 Ada GPUs for 2.5 days. The learning rate is set to 2.5e-5, with a per-device training batch size of 8 and gradient accumulation steps of 4. The total number of training epochs is 4. We employ Low-Rank Adaptation (LoRA) (Hu et al., 2021) during training, with 4-bit quantization. The paged Adam optimizer (Kingma, 2014), adapted for quantization, is used to further optimize the training process. To prepare the inputs for training, we compute the largest token length in our dataset and create custom tokenization with left padding. We set the End-of-Sequence token as the pad token to ensure compatibility with causal language models.

4 Evaluation Methodology

4.1 Baseline Techniques

Baselines without LLM. ACT (Gupta et al., 2022) and CaML (Balaji et al., 2023) are model-based carbon estimation techniques that do not rely on large

language models (LLMs). ACT calculates the carbon footprint of each component within computer systems using detailed product manufacturing information. On the other hand, CaML associates product names with North American Industry Classification System (NAICS) codes to estimate a carbon footprint per dollar at the industry sector level. Given that CaML consistently provides the same estimate for 'Electronic Computer Manufacturing,' we assume a default price for computing products to calculate the overall carbon footprint. These models are evaluated by estimating and comparing the total carbon footprint of products against ground truth values.

Baselines without RAG We use Gemini-1.5-flash (Google DeepMind, 2023) and Llama 3_{8B} (Meta AI, 2023) as our baselines without RAG. These powerful open-source LLMs are included to analyze the carbon reasoning capabilities of LLMs without data augmentation.

Baselines with RAG We evaluate Self-RAG (Asai et al., 2023), TAT-LLM (Zhu et al., 2024), Gemini-1.5-flash and GPT-4 (OpenAI, 2023) as baselines with RAG. Self-RAG retrieves relevant documents and guides the LLM to generate the best possible answer. TAT-LLM emphasizes using LLMs to answer questions based on well-formatted tables and texts. Additionally, we provide the exact reference text along with the question to Gemini-1.5-flash and GPT-4 to assess their performance on CarbonPDF-QA dataset without fine-tuning.

4.2 Metrics

We adopt the Mean Squared Error (MSE) and Mean Absolute Error (MAE) to measure the numerical accuracy of the predicted answers from the gold answers. We also use Exact Match (EM) to measure how often the predicted values match the gold exactly (Rajpurkar et al., 2016). For questions with multiple answers, the model is required to match all gold answers exactly, including their order, to be considered correct.

5 Results

5.1 Baseline Performance

Table 2 compares CarbonPDF with other baseline techniques. Our technique consistently outperforms all baselines. Model-based approaches such as ACT and CaML show high MSE and MAE due to their reliance on general carbon estimates and default values, which lack customization for specific questions. Consequently, their Exact Match (EM) scores are close to zero. LLM baselines without RAG — Gemini-1.5-flash and Llama 3_{8B} — also exhibit significant errors, with EM values below 10%. This underscores the need for data augmentation to improve the accuracy of model predictions.

While RAG-based approaches show improved performance, they still have lower performance compared to our model. GPT-4 outperforms Gemini-1.5-flash with higher EM and lower MAE values, though it still makes significant errors on some questions. We also compared our technique to Self-RAG, which uses text to answer questions. However, the technique struggles with complex reasoning questions. Additionally, our model surpasses reasoning-based models like TAT-LLM, which rely on well-formatted tables and text input. This highlights the challenges current QA models face in handling complex and inconsistent data in CarbonPDF-QA dataset.

Table 2: Baseline performance comparison.

Techniques	MSE	MAE	EM
Baselines without LLM			
ACT (Gupta et al., 2022)	2.37e5	323.80	0.00
CaML (Balaji et al., 2023)	1.90e5	230.70	0.28
Baselines without RAG			
Gemini-1.5-flash	7.40e6	163.49	9.56
Llama 3_{8B}	1.45e33	1.18e14	5.23
Baselines with RAG			
Self-RAG (Asai et al., 2023)	7.22e6	173.03	16.79
TAT-LLM (Zhu et al., 2024)	9.96e9	2584.14	0.152
Gemini-1.5-flash	4.35e6	111.03	28.50
GPT-4	1.25e4	33.01	51.47
CarbonPDF	**81.02**	**0.35**	**98.48**

Table 3: CarbonPDF performance on different question types

Type	#Question	MSE	MAE	EM
Word Match	1959	2.35	0.08	98.11
Max/Min	487	0	0	100.00
Top 3/5	369	0.03	0.01	99.46
Calculation	2062	189.39	0.74	98.30

5.2 Performance on Different Question Types

Table 3 summarizes our CarbonPDF performance across different question types shown in Figure 1. The model performs well across all these question types. Max/min and top-3/5 questions yield slightly better results, likely due to their simpler reasoning

Table 4: Performance on multi-answer questions

#Answer	#Question	MSE	MAE	EM
0	1088	0.00	0.00	100.00
1	1165	71.31	0.38	98.97
2	878	81.49	0.50	98.41
3	849	54.97	0.32	97.88
4	712	272.26	0.75	95.93
5	185	1.08e-5	1.08e-4	99.46

Table 5: Ablation analysis of CarbonPDF.

Task	MSE	MAE	EM
Few-shot	7.47e6	193.54	10.29
Program-based Reasoning	3.19e3	3.65	75.35
CarbonPDF	**81.02**	**0.35**	**98.48**

and fewer question variants, which reduce the potential for errors. Word matching questions exhibit similar EM compared to calculation questions but have much lower MSE values in comparison. This is because word-matching questions require CarbonPDF to extract relevant evidence and provide an answer, whereas calculation questions involve more complex reasoning, increasing the potential for errors.

5.3 Performance on Multi-answer Questions

We now analyze the impact of questions requiring multiple answers. Table 4 shows the results as we vary the number of answers per question. For questions with irrelevant references, where no answer is expected, our model accurately identifies these references and provides no answers. As the number of required answers increases, the Exact Match (EM) score gradually decreases due to the added complexity in evidence extraction and carbon modeling. Most multi-answer questions involve a mix of calculation and word-matching types, which can reduce accuracy. However, questions requiring five answers still perform well because they primarily consist of top-5 question types that CarbonPDF handles effectively. In summary, CarbonPDF performs well across questions with varying numbers of answers. Nonetheless, complex questions with fewer answers tend to show better performance.

5.4 Ablation Study

We conduct an ablation study to evaluate the impact of various components. First, we analyze the effectiveness of few-shot learning in our pipeline. Few-shot learning involves providing a small number of examples at inference time to guide the desired completion, which has been shown to perform well in some tasks (Brown, 2020; Gautier et al., 2022). Thus, we replace the fine-tuning step in CarbonPDF with a few-shot approach, where we provide two examples to derive the final result. Table 5 shows the results of this approach. We observe that the few-shot technique does not perform well on our dataset. This is consistent with prior work that indicates few-shot methods struggle with complex reasoning tasks (Brown, 2020; Asai et al., 2023).

In addition to evaluating the few-shot approach, we also evaluated CarbonPDF without the program-based reasoning step. In this variation, we trained CarbonPDF to generate the final answer directly without using the program. This approach showed improved performance compared to the few-shot technique. However, even with this modification, the performance did not surpass the CarbonPDF model.

Table 6: Errors sources in CarbonPDF.

Error Type	Number	Percentage %
Evidence	74	100.00
Program Reasoning	53	71.62
Relevant Token	23	31.08
PDF File Parsing	9	12.16

5.5 Error Analysis

We now analyze the sources of error in CarbonPDF's outputs. To identify these errors, we analyze the questions that were answered incorrectly and determine the underlying causes. We classify errors into four categories: (i) Evidence Errors, where the evidence index or text retrieved by the system differs from the ground truth, indicating that incorrect evidence was identified; (ii) Program Reasoning Errors, which occur when the program's execution results in incorrect outputs due to flaws in logic or reasoning; (iii) Relevant Token Errors, where the predicted relevant tokens do not align with the ground truth, often leading to irrelevant text being incorrectly selected as relevant; and (iv) PDF Parsing Errors, which arise from inconsistencies or spurious data during the PDF extraction process, resulting in inaccuracies. Note that an output may belong to multiple error categories.

Table 6 provides a summary of the errors in CarbonPDF, including their respective counts and percentages. We observe that all the incorrect answers have issues with extracting the evidence. These errors often occur due to the large size of the PDF

document and the separation of name-value pairs across different locations, making it more challenging to accurately locate the relevant evidence. The second largest source of errors is program generation, which is impacted by all other errors since it is the final step in the process. Additionally, 31% of the incorrect answers result from wrongly predicted relevant tokens. This error often occurs due to the complexity and size of unstructured PDF documents, making it difficult for the model to understand and identify the relevant keywords. Approximately one-tenth of the errors are related to PDF file parsing, which can lead to misinterpretations during evidence location or carbon modeling. Detailed examples for each error type are discussed in Appendix A.3.

6 Related Work

QA Datasets There are numerous existing QA datasets. For structured data, such as Knowledge Base (KB) and tables, notable examples include Complex Web Questions (Talmor and Berant, 2018) and TabFact (Chen et al., 2019). Text-based QA datasets include SQuAD (Rajpurkar et al., 2016), SearchQA (Dunn et al., 2017), and DROP (Dua et al., 2019). For multi-hop QA, there are HOTPOTQA (Yang et al., 2018) and HybridQA (Chen et al., 2020). Hybrid datasets also include TAT-QA (Zhu et al., 2021), which integrates tabular and textual content in the financial domain, and TAT-LLM (Zhu et al., 2024), which utilizes well-formatted tabular and textual data to train LLMs on discrete reasoning. Our CarbonPDF-QA dataset stands apart by including inconsistent or spurious data extracted from PDFs, reflecting the challenges of real-world document processing. Furthermore, the tables in our dataset are not well-structured, with column values that may span different paragraphs, complicating data analysis.

QA Reasoning Numerous studies have explored question-answering (QA) systems, including those that use Retrieval-Augmented Generation (RAG) approaches to guide large language models (LLMs) in answering questions (Wei et al., 2022; Gao et al., 2023; Guu et al., 2020; Lewis et al., 2020; Asai et al., 2023). Despite these advancements, LLMs often struggle with complex reasoning tasks, particularly numerical reasoning. Recent research has focused on numerical reasoning over tabular and textual data (Zhu et al., 2021; Li et al., 2022; Zhu et al., 2022; Zhou et al., 2022; Li et al., 2023; Wei et al., 2023), including financial reports (Chen et al., 2021; Yuan et al., 2024). However, the application of these techniques to real-world hybrid data, such as that extracted from PDFs, remains relatively unexplored. Our work addresses this gap by focusing on the challenges posed by inconsistent data in the context of sustainability reports.

Carbon Footprint Analysis Companies frequently employ Lifecycle Assessment (LCA) methodologies to evaluate the environmental impact of their products across the entire lifecycle, from raw material extraction to disposal (Hauschild et al., 2018). Tools like GaBi and SimaPro are widely used for conducting these assessments, producing detailed analyses that are often integrated into sustainability reports (Silva et al., 2017). However, LCA methods require significant manual effort and depend heavily on detailed input data, which companies often do not publicly disclose. Recent advancements have focused on automating carbon footprint analysis through data-driven approaches (Gupta et al., 2022; Balaji et al., 2023). These approaches typically utilize publicly available data, such as industry averages or estimates, which tend to be less accurate than the more precise, company-specific data used in traditional LCA methods. The application of question-answering (QA) systems within the sustainability domain is relatively nascent. To the best of our knowledge, our work is the first to apply QA systems for carbon footprint assessments within sustainability reports.

7 Conclusion

In this work, we introduce CarbonPDF-QA, an open-source product carbon footprint QA dataset with comprehensive annotations, comprising around 25,000 questions of various types. We leverage this dataset to fine-tune CarbonPDF, enabling it to perform reasoning with reference augmentation and generate accurate results through our program-based reasoning approach. We demonstrate its effectiveness through extensive experiments and show that CarbonPDF outperforms the best baseline on all metrics. We anticipate that the CarbonPDF-QA dataset and CarbonPDF model will serve as valuable benchmarks and baselines, fostering the development of more advanced QA models for PDF documents and carbon footprint estimation.

8 Limitations

One key limitation of current PDF parsing methods is their difficulty in handling data presented in graphical forms, such as pie charts or bar graphs. These visual elements are often used to convey complex data, but traditional parsing techniques that focus on text extraction struggle with purely graphical content. This limitation poses a significant challenge, as crucial information within these visual elements can be missed or misinterpreted. Since CarbonPDF primarily relies on text data, it cannot effectively answer questions based on content that combines graphs and text. However, our technique remains useful when numerical data is presented alongside these graphs, as it can still extract and analyze this information. In the future, we plan to explore multimodal large language models (LLMs) to perform reasoning on both text and visual data.

Although CarbonPDF can handle various types of questions, there are still limitations. For example, if CarbonPDF is asked about the carbon footprint of processors, but the exact term "processor" does not appear in the text, the system might incorrectly deem the reference as irrelevant, even if related terms like "mainboard" are present. This occurs because the model is not capable of understanding synonyms or recognizing that certain components are subsets of larger systems. A key question for future research is whether large language models (LLMs) can be trained to handle such nuances, improving their reasoning ability to understand related terms and components within a broader context.

9 Ethics Statement

In this work, we first highlight the challenges of processing PDF documents using examples from our dataset. We then discuss how we collected and processed our CarbonPDF-QA dataset. Following this, we propose our CarbonPDF model, which finetunes an LLM (Llama 3_{8B}) to perform program-based reasoning on unstructured PDF data. Our model is developed using open-source tools and datasets to aid in understanding and processing of product sustainability reports. Therefore, we do not anticipate any potential risks or negative ethical issues associated with this work.

9.1 Data Collection and Licensing

We collected the product carbon footprint reports that are publicly available. Our CarbonPDF-QA dataset consists of content extracted from these reports. We plan to release the data under CDLA-Permissive[1] license. This will allow broad access and use of our dataset, allowing recipients to modify and share the data freely.

9.2 Potential Risk

CarbonPDF may produce inaccurate results for certain questions. This may lead to misleading conclusions or errors in evaluating environmental impacts.

References

Acer Inc. 2024. Acer library document collection (product carbon footprint). Accessed: 2024-08-12.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Self-reflective retrieval augmented generation. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Bharathan Balaji, Venkata Sai Gargeya Vunnava, Geoffrey Guest, and Jared Kramer. 2023. Caml: Carbon footprinting of household products with zero-shot semantic text similarity. In *Proceedings of the ACM Web Conference 2023*, pages 4004–4014.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, et al. 2021. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. Convfinqa: Exploring the chain of numerical reasoning in conversational finance question answering. *arXiv preprint arXiv:2210.03849*.

Dell Inc. 2024. Dell library document collection (product carbon footprint). Accessed: 2024-08-12.

[1] https://cdla.dev/permissive-2-0/

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Izacard Gautier, Lewis Patrick, Lomeli Maria, Hosseini Lucas, Petroni Fabio, Schick Timo, Dwivedi-Yu Jane, Joulin Armand, Riedel Sebastian, and Grave Edouard. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv: 2208.03299*.

Google DeepMind. 2023. Gemini-1.5-flash. Accessed: 2024-08-10.

Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S Lee, David Brooks, and Carole-Jean Wu. 2022. Act: Designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 784–799.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Michael Z Hauschild, Ralph K Rosenbaum, Stig Irving Olsen, et al. 2018. *Life cycle assessment*, volume 2018. Springer.

HP Inc. 2024. HP library document collection (product carbon footprint). Accessed: 2024-08-12.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

DP Kingma. 2014. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lenovo Inc. 2024. Lenovo library document collection (product carbon footprint). Accessed: 2024-08-12.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Moxin Li, Fuli Feng, Hanwang Zhang, Xiangnan He, Fengbin Zhu, and Tat-Seng Chua. 2022. Learning to imagine: Integrating counterfactual thinking in neural discrete reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 57–69.

Xiao Li, Yin Zhu, Sichen Liu, Jiangzhou Ju, Yuzhong Qu, and Gong Cheng. 2023. Dyrren: A dynamic retriever-reranker-generator model for numerical reasoning over tabular and textual data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13139–13147.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Meta AI. 2023. Llama 3 8b. Accessed: 2024-08-10.

Olivier M. Schwab. 2022. Sustainability reporting: why it's important. [Online; accessed 14-August-2024].

OpenAI. 2023. GPT-4. `https://openai.com/research/gpt-4`. Accessed: 2024-08-12.

PyMuPDF Developers. 2024. PyMuPDF documentation. Accessed: 2024-08-08.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Rodriguez, Isabel and Caglio, Ariela. 2023. Tackling the sustainability reporting challenge. a policy guide. [Online; accessed 14-August-2024].

Diogo Silva, A Oliveira Nunes, Aparecida da Silva Moris, Cassiano Moro, and Thiago Oliveira Rodrigues Piekarski. 2017. How important is the lca software tool you choose comparative results from gabi, openlca, simapro and umberto. In *Proceedings of the VII Conferencia Internacional de Análisis de Ciclo de Vida en Latinoamérica, Medellin, Colombia*, pages 10–15.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

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

Yifan Wei, Fangyu Lei, Yuanzhe Zhang, Jun Zhao, and Kang Liu. 2023. Multi-view graph representation learning for answering hybrid numerical reasoning question. *arXiv preprint arXiv:2305.03458*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Ziqiang Yuan, Kaiyuan Wang, Shoutai Zhu, Ye Yuan, Jingya Zhou, Yanlin Zhu, and Wenqi Wei. 2024. Finllms: A framework for financial reasoning dataset generation with large language models. *arXiv preprint arXiv:2401.10744*.

Yongwei Zhou, Junwei Bao, Chaoqun Duan, Youzheng Wu, Xiaodong He, and Tiejun Zhao. 2022. Unirpg: Unified discrete reasoning over table and text as program generation. *arXiv preprint arXiv:2210.08249*.

Fengbin Zhu, Wenqiang Lei, Fuli Feng, Chao Wang, Haozhou Zhang, and Tat-Seng Chua. 2022. Towards complex document understanding by discrete reasoning. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4857–4866.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *arXiv preprint arXiv:2105.07624*.

Fengbin Zhu, Ziyang Liu, Fuli Feng, Chao Wang, Moxin Li, and Tat-Seng Chua. 2024. Tat-llm: A specialized language model for discrete reasoning over tabular and textual data. *arXiv preprint arXiv:2401.13223*.

A Appendix

A.1 Prompt templates

The template for CarbonPDF training prompt is shown in Listing 1. For testing, we use the prompt in Listing 2 and ask the model to complete it. List 3 displays the few-shot prompt template used to obtain the results in Table 5. List 4 and List 5 are the prompt templates designed to train and test the model without carbon modeling.

A.2 Evidence Locating Examples

Table 7 presents examples of evidence locating during the testing of CarbonPDF across different question types. Generally, CarbonPDF successfully identifies the correct evidence text, though it occasionally makes mistakes in predicting the precise evidence index (with blue text representing the ground truth and red text indicating errors in prediction). However, since the carbon modeling step relies on the text itself rather than the index, our goal of guiding the LLM to find the evidence locations is effectively achieved. It is also worth noting that in the Word Match example, there are two start-end index pairs for each evidence text. This occurs because the percentage value and the associated name are separated into different positions within the text, requiring the use of two start-end pairs to accurately locate them.

A.3 Carbon PDF Error Examples

Table 8 provides examples of errors in CarbonPDF. In the evidence locating error example, the prediction incorrectly identifies a percentage of 7.1% (highlighted in red) instead of the correct value of 14.8% (highlighted in blue) located just before it, leading to an incorrect final answer. In the program generation error example, the LLM generates two unnecessary lines of code, which result in the correct carbon footprint of the power supply unit being incorrectly calculated as the total carbon footprint. In the example for relevant token prediction error, although both the product name and the components are present in the reference, CarbonPDF incorrectly classifies it as irrelevant. For the PDF file parsing error example, the blue text in the corrected reference indicates where the red text in the original raw reference should be. After converting a PDF file into raw text, issues such as spurious data (like the duplicated "0.0%") and formatting problems (such as the long space between "g" and "e" in "percentage") can be introduced, leading to misinterpretations.

Listing 1: CarbonPDF Training Prompt Template

```
You'll be provided with some questions and a reference. First, you must check
    whether the reference is relevant to the question and generate a token. If the
    reference is relevant, identify the necessary evidence from it to answer the
    questions. View the whole reference text as a character array. Output the
    evidence and the locations of the evidence as start and end indexes in the
    character array. Based on the evidence and indexes, generate the Python program
    to compute and answer the questions. The indexes are enclosed by square brackets
    . The program is enclosed by triple backticks. The final answer in the program
    is of list type.
### Question: {question}
### Reference: {reference text}
### Is Reference Text Relevant?: {True/False}
### Evidence: {[index]:"evidence text"}
### Program:
```
{program}
```
```

Listing 2: CarbonPDF Testing Prompt Template

```
You'll be provided with some questions and a reference. First, you must check
    whether the reference is relevant to the question and generate a token. If the
    reference is relevant, identify the necessary evidence from it to answer the
    questions. View the whole reference text as a character array. Output the
    evidence and the locations of the evidence as start and end indexes in the
    character array. Based on the evidence and indexes, generate the Python program
    to compute and answer the questions. The indexes are enclosed by square brackets
    . The program is enclosed by triple backticks. The final answer in the program
    is of list type.
### Question: {question}
### Reference: {reference text}
### Is Reference Text Relevant?:
```

Question Type	**Evidence**
Word Match	G:{"[1014,1018], [1643,1649]": "47.7%, Display", "[1032,1035], [1684,1703]": "7.1%, Power Supply Unit(s)", "[1062,1065], [1809,1817]": "0.5%, Packaging"} P:{"[1014,1018], [1643,1649]": "47.7%, Display", "[1030,1033], [1684,1703]": "7.1%, Power Supply Unit(s)", "[1059,1062], [1809,1817]": "0.5%, Packaging"}
Max/Min	G:{"[2396,2410]": "Hard Drive 3.0%", "[2430,2441]": "Battery 2.7%", "[2372,2394]": "Chassis & Assembly 4.8%", "[2412,2428]": "Power Supply 7.0%", "[2443,2474]": "Mainboard and Other Boards 26.6%", "[2476,2488]": "Display 35.2%", "[2490,2503]": "Packaging 0.4%"} P:{"[2396,2410]": "Hard Drive 3.0%", "[2430,2441]": "Battery 2.7%", "[2372,2394]": "Chassis & Assembly 4.8%", "[2412,2428]": "Power Supply 7.0%", "[2443,2474]": "Mainboard and Other Boards 26.6%", "[2476,2488]": "Display 35.2%", "[2490,2503]": "Packaging 0.4%"}
Top 3/5	G:{"[138,150]": "Chassis 21.0%", "[180,219]": "Power Supply Unit & External Cables 1.7%", "[106,136]": "Mainboard and other boards 9.6%", "[92,104]": "Display 61.6%", "[165,178]": "Packaging 2.2%"} P:{"[138,150]": "Chassis 21.0%", "[180,219]": "Power Supply Unit & External Cables 1.7%", "[106,136]": "Mainboard and other boards 9.6%", "[92,104]": "Display 61.6%", "[165,178]": "Packaging 2.2%"}
Calculation	G:{"[2925,2938]": "125kg CO 2 eq.", "[2944,2960]": "Manufacturing 69%", "[211,248]": "Power Supply Unit & External Cables 4%", "[148,158]": "Display 33%", "[187,196]": "Chassis 7%", "[117,146]": " Mainboard and other boards 38%"} P:{"[2925,2938]": "125kg CO 2 eq.", "[2944,2960]": "Manufacturing 69%", "[211,248]": "Power Supply Unit & External Cables 4%", "[147,157]": "Display 33%", "[187,196]": "Chassis 7%", "[117,146]": " Mainboard and other boards 38%"}

Table 7: Examples of the evidence locating in our CarbonPDF. G and P represent ground truth and prediction.

Listing 3: Few-Shot Prompt Template

```
You'll be provided with some questions and a reference. First, you must check whether the reference is relevant to the question and generate a token. If the reference is relevant, identify the necessary evidence from it to answer the questions. View the whole reference text as a character array. Output the evidence and the locations of the evidence as start and end indexes in the character array. Based on the evidence and indexes, generate the Python program to compute and answer the questions. The indexes are enclosed by square brackets. The program is enclosed by triple backticks. The final answer in the program is of list type.
Here are some examples.

Example 1:
### Question: {question}
### Reference: {reference text}
### Is Reference Text Relevant?: {True}
### Evidence: {[index]:"evidence text"}
### Program:
```
{program}
```

Example 2:
### Question: {question}
### Reference: {reference text}
### Is Reference Text Relevant?: {False}
### Evidence: {[index]:"evidence text"}
### Program:
```
{program}
```

Now the question and reference are shown below. What are the answers to the question?
### Question: {question}
### Reference: {reference text}
### Is Reference Text Relevant?:
```

Listing 4: CarbonPDF without Carbon Modeling Training Prompt Template

```
You'll be provided with some questions and a reference. First, you must check whether the reference is relevant to the question and generate a token. If the reference is relevant, extract the essential information to answer the questions. View the whole reference text as a character array. Output the evidence and the locations of the evidence as start and end indexes in the character array. Based on the evidence and indexes, compute and answer the questions. The indexes are enclosed by square brackets. The final answer is of list type.
### Question: {question}
### Reference: {reference text}
### Is Reference Text Relevant?: {True/False}
### Evidence: {[index]:"evidence text"}
### Answer: {answer}
```

Listing 5: CarbonPDF without Carbon Modeling Testing Prompt Template

```
You'll be provided with some questions and a reference. First, you must check whether the reference is relevant to the question and generate a token. If the reference is relevant, extract the essential information to answer the questions. View the whole reference text as a character array. Output the evidence and the locations of the evidence as start and end indexes in the character array. Based on the evidence and indexes, compute and answer the questions. The indexes are enclosed by square brackets. The final answer is of list type.
### Question: {question}
### Reference: {reference text}
### Is Reference Text Relevant?:
```

Error Type	Example
Evidence	Q: What are the carbon footprint percentages of the manufacturing, power supply unit, and mainboard and other boards in the VS2690G desktop? R: 23.8% 14.8% 7.1% ... Product breakout ... Mainboard (and other boards) Power Supply Unit(s) Chassis.. Manufacturing 53.4% G: [53.4%, 14.8%, 23.8%] P: [53.4%, 7.1%, 23.8%]
Program Reasoning	Q: What are the carbon footprints of power supply unit, total, and batteries in the C723T-TCO laptop? R: 212 +/- 40†kgCO2e ... 9.5% ... 3.4% ... Power Supply Unit(s) ... Battery G Program: total_carbon=212.0 power_percent=0.095 power_carbon=total_carbon*power_percent batteries_percent=0.034 batteries_carbon=total_carbon*batteries_percent answer=[power_carbon,total_carbon,batteries_carbon] P Program: total_carbon=212.0 power_percent=0.095 power_carbon=total_carbon*power_percent power_answer=total_carbon batteries_percent=0.034 batteries_carbon=total_carbon*batteries_percent batteries_answer=batteries_carbon answer=[power_answer,total_carbon,batteries_answer] G: [20.14, 212.0, 7.208] P: [212.0, 212.0, 7.208]
Relevant Token	Q: What are the carbon footprints of HDD, display, and batteries in the Latitude 3520 laptop? R: Dell Latitude 3520 ... Hard Drive 3.1% ... Battery 2.6% ... Display 42.8% G: True P: False
PDF File Parsing	Q: What are the carbon footprint percentages of the packaging, mainboard and other boards, chassis, and power supply unit in the VX8715GT desktop? R: by percentag e % 43.0% 23.9% 15.8% 7.5% 3.0% 2.0% 1.7% 1.7% 0.9% 0.6% 0.0% 0.0% ... End of Life Product breakout Use Mainboard (and other boards) Power Supply Unit(s) Chassis Transport Optical Drive(s) Hard Drive(s) Solid State Drive(s) 2023/Jun Packaging 0.00 0.00 Product Weight R Corrected: by percentage % 43.0% 23.9% 15.8% 7.5% 3.0% 2.0% 1.7% 1.7% 0.9% 0.6% ~~0.0% 0.0%~~ ... Product breakout Use Mainboard (and other boards) Power Supply Unit(s) Chassis Transport Optical Drive(s) Hard Drive(s) Solid State Drive(s) End of Life Packaging ~~0.00 0.00~~ 2023/Jun Product Weight G:[0.6%, 23.9%, 7.5%, 15.8%] P:[1.7%, 23.9%, 7.5%, 15.8%]

Table 8: Examples of the errors in our CarbonPDF. Q, R, G, and P represent question, reference text, ground truth, and prediction.