Extracting Greenhouse Gas Emission Values from Corporate Sustainability Reports

A Case Study on Evaluating and Finetuning Language Models on Long-Context Structured Information Extraction

February 1, 2024

Abstract

Information about greenhouse gas emissions by corporations is useful for a variety of purposes. However, this information is not published in a machine-readable format. Interested actors have to manually extract this information from sustainability reports. To solve this, a retrieval-augmented generation (RAG) system is developed to use language models for the long-context task of extracting scope 1, 2 and 3 emission data from sustainability reports. Furthermore, since ground truth data about this task is not publicly available, a dataset of sustainability reports and manually-extracted emission values ($N = 100$) is collected. The collected data and developed system is used to evaluate a selection of language models, with Mixtral-8x7B-Instruct-v0.1 performing the best overall.

Using this information, a synthetic finetuning dataset is created using 3233 sustainability reports and emission data extracted by Mixtral-8x7B-Instruct-v0.1. Finetuning a LoRA for the significantly smaller Mistral-7B-Instruct-v0.2 on this dataset leads to a model that nearly reaches the performance of Mixtral-8x7B-Instruct-v0.1 on emission value extraction with an accuracy of 65% (up from 46% of the base model) and exceeds its performance on source page citation with an accuracy of 77% (base model: 53%), while being only a sixth of its size.

The findings of this work provide general indications on using language models for long-context structured information extraction tasks. The source code, data and models created in this project are open-sourced at `https://github.com/nopperl/corporate_emission_reports`.

1 Problem Statement

Information about greenhouse gas emissions by corporations is useful for a variety of actors including journalists, policymakers and civil society. The importance of this information is underlined by the adoption of the Corporate Sustainability Reporting Directive [1], which requires EU companies to publish this

[1]`https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32022L2464`

information as part of sustainability reports starting with 2025. However, these reports are usually published as PDF documents and not machine-readable. Hence, emission information has to be extracted manually, which is a tedious and time-consuming task. To solve this problem, the aim of this project is to develop and evaluate a machine learning system to automatically extract this information from sustainability reports. The output can then be easily used by interested actors for further data analysis, e.g. as foundation for media reporting on climate change.

Furthermore, to evaluate this system and to encourage research into solving such a task, a dataset of sustainability reports and manually-extracted emission values is created and published.

As the dataset and system should be publically-accessible and inspectable, the focus of this project is on open-source models and software and the code and dataset are published at `https://github.com/nopperl/corporate_emission_reports`.

1.1 Research objective

The main result is a machine learning system which, given a sustainability report, extracts the Scope 1, 2 and 3 greenhouse gas emissions in metric tonnes of CO2eq in a machine-readable format. For the purpose of this project, the following definitions from the GHG Protocol Corporate Standard[2] are used:

- Scope 1: A reporting organization's direct GHG emissions.
- Scope 2: A reporting organization's emissions associated with the generation of electricity, heating/cooling, or steam purchased for own consumption.
- Scope 3: A reporting organization's indirect emissions other than those covered in scope 2.

Note, that for scope 2 emissions, the market-based calculation is given precedence.

Furthermore, the system indicates if this data is missing from the report. To aid interpretability, the system also cites the parts of the input report referenced during extraction.

The second result is a dataset of corporate sustainability reports with manually-extracted emission values, which can be used to benchmark and evaluate automatic extraction systems.

The third result is an evaluation of the performance of the developed system using different language models based on the collected dataset.

[2]`https://ghgprotocol.org/corporate-standard`

2 Dataset

To develop and evaluate the intended information extraction system, a gold standard dataset of corporate sustainability reports and their associated scope 1, 2 and 3 emissions is created. Since there is no publicly-available self-reported greenhouse gas emissions dataset, these values have to be manually extracted from the sustainability reports. Hence, the dataset is restricted to a small size of $N = 100$, which is not large enough for training or finetuning models, but suffices for evaluation. For simplicity, only reports in English are considered. Note that sustainability reports based on the Carbon Disclosure Project [3] or Global Reporting Initiative [4] templates are not considered in order not to overfit to a specific format.

To ensure geographic diversity, three sets of companies headquartered on different continents are considered for the dataset. The first set consists of 39 corporations tracked by the Euro Stoxx 50 stock index as of 18 September 2023. Note that out of the missing 11 corporations of Euro Stoxx 50, 5 are not considered as no publicly-available sustainability reports were found and 6 are not considered as the author of this project was unable to extract unambiguous emission values. The second set is a random selection of 39 corporations listed on the New York Stock Exchange as of December 2023. The third set is a random selection of 22 corporations tracked by the Nikkei 225 index as of October 2023. Note, that this selection strategy is intentionally biased towards larger corporations due to the assumption that they have a higher likelihood of publishing sustainability reports.

For every corporation, the most recent sustainability report is downloaded from the official source. In some cases, the sustainability report is part of a larger annual report. The following information is manually extracted by the author of this project:

- Scope 1 emissions in metric tonnes of CO2eq
- Scope 2 emissions in metric tonnes of CO2eq
- Scope 3 emissions in metric tonnes of CO2eq
- List of pages containing Scope 1 data
- List of pages containing Scope 2 data
- List of pages containing Scope 3 data

If the data does not exist in the report or cannot be unambiguously extracted, it is noted as missing.

Furthermore, additional data of potential use for future projects is extracted. These consists of:

[3] `https://www.cdp.net/en/guidance`
[4] `https://www.globalreporting.org/standards/`

Model	**Param Size**	**Context Length**
Mistral-7B-Instruct-v0.2 [3]	7B	32768
openchat-3.5-0106 [9]	7B	8192
Qwen-1.8B-Chat [1]	1.8B	8192
Mixtral-8x7B-Instruct-v0.1	45B	32768
miqu-1-70b	70B	32764

Table 1: Language models used for the extraction task.

- Year of the most recent emission data
- Emissions for each of the 15 scope 3 categories [5] in metric tonnes of CO2eq
- Report URL
- SHA-256 hash of the report PDF

3 System

The system to automatically extract emission values from sustainability reports consists of four parts: The *input data* (i.e., a sustainability report), which is inserted into a *prompt*, which is used by a *language model* to produce an *output* (i.e., the structured emission data).

3.1 Model

The backbone of this system is the machine-learning model. Due to their versatility, decoder-only generative language models are suited for this task. Since the focus of this project is on open source models, proprietary models such as the state-of-the-art GPT-4 [7] are not considered.

There are usually three types of models: base models, finetuned models and aligned models. Base models are simply trained on a large dataset of diverse texts. These base models can then be tailored for instruction-following tasked by supervised fine-tuning (SFT) on a smaller instruction dataset. Even further, these models can then be aligned to human preference using direct preference optimization (DPO) [8] or reinforcement learning through human feedback. Since the instruction following capability is important to achieve the task, only instruction-following models are considered.

Table 1 gives an overview of the language models used in this project. Out of the *small* class of language models consisting of 7 billion parameters or lower, Mistral can be considered state-of-the-art with its remarkable general performance [3]. Note, that the instruction-tuned version of Mistral is used.

Keep in mind that language models are trained on a maximum sequence length, also known as context size. Mistral is trained for an unusually large

[5] `https://ghgprotocol.org/scope-3-calculation-guidance-2`

context size of 32768 tokens. Since the task contains long input reports, this makes Mistral uniquely suited. Preliminary tests using different models with a context size of 2048 or 4096 showed significantly worse performance.

In order to investigate how much a larger training context size affects performance, the openchat model is used as alternative in the 7b class. While it was trained with a sequence length of only 8192 tokens, using the recent self-extend technique [4] it might be competitive with Mistral on the studied task.

Furthermore, to investigate how large the parameter size needs to be for this task, the significantly smaller Qwen-1.8B model is also evaluated. Similarly to openchat-3.5-0106, this model was trained with a sequence length of 8192 tokens and requires the self-extend technique to work.

To ascertain the upper limit, the significantly larger Mixtral and miqu are also evaluated. These two models are selected is used as they performed significantly better than other ≥ 45B models in preliminary experiments, likely due to their larger context size.

All models are run using the llama.cpp [6] inference engine.

3.2 Input

The input data consists of a sustainability report. While the report is in PDF format, language models operate on plain text. Hence, a plain-text XHTML representation of the PDF document is extracted using PyMuPDF [7]. Importantly, images are stripped from this representation. While it is also possible to purely extract text content using PyMuPDF, the XHTML representation is chosen as it preserves some logical structure of the PDF document. An important example of this are footnotes, which can be placed right next to the emission values and thus make it impossible for the model to discern the correct value.

As sustainability reports are usually longer than a few pages, the entirety of their text does not fit into the context of language models. This problem is solved using retrieval-augmented generation (RAG) [5]. That is, the report is split into chunks and only chunks relevant to the input query are included in the generation prompt.

There are two important questions in RAG: how to split the input into chunks and how to find relevant chunks to the query. In this project, simple solutions are used for both. The input report is split into chunks by pages, i.e. every page is a chunk. An entire page is an unusually large chunk, but it represents a logical boundary that avoids the problem of smaller chunking leaving semantically associated information in different chunks.

The second task, i.e. finding relevant chunks, is usually done by embedding the chunks and input query into vector space using an embedding model such as UAE-Large-V1 [6] and then selecting the chunks with the highest vector similarity. However, a simpler alternative is available in this project, as the

[6]`https://github.com/ggerganov/llama.cpp`
[7]`https://pymupdf.readthedocs.io/en/latest/index.html`

relevant chunks contain the somewhat unique terms `Scope 1`, `Scope 2` and `Scope 3`. Simply selecting only chunks containing those terms is a crude but sufficient and fast solution.

3.3 Prompt

As the system uses general-purpose language models, a well-engineered prompt is crucial in order to produce the intended output. Furthermore, a good prompt is likely to boost the performance of the relatively small language models considered in this work.

An instruction-following prompt in ChatML format is used for this work, i.e. the prompt simulates a conversation between a user and an assistant and induces the model to assume the role of the assistant in its generation. It can be seen in Listing 2 and is very detailed in order to prevent unintended outputs observed during development.

For every report, the instruction prompt is filled with the extracted chunks. Due to this, the model inputs reach a considerable length, as can be seen in Table 2. As this dangerously approaches (and sometimes even exceeds) the model context limits, promising but token-expensive prompting techniques such as chain-of-thought [11] or few-shot [2] prompting can not be utilized.

	max	mean	median	min
token_length	60063.00	14544.31	12184.50	1004.00

Table 2: Statistics about the number of tokens in input prompts tokenized for the Mistral model.

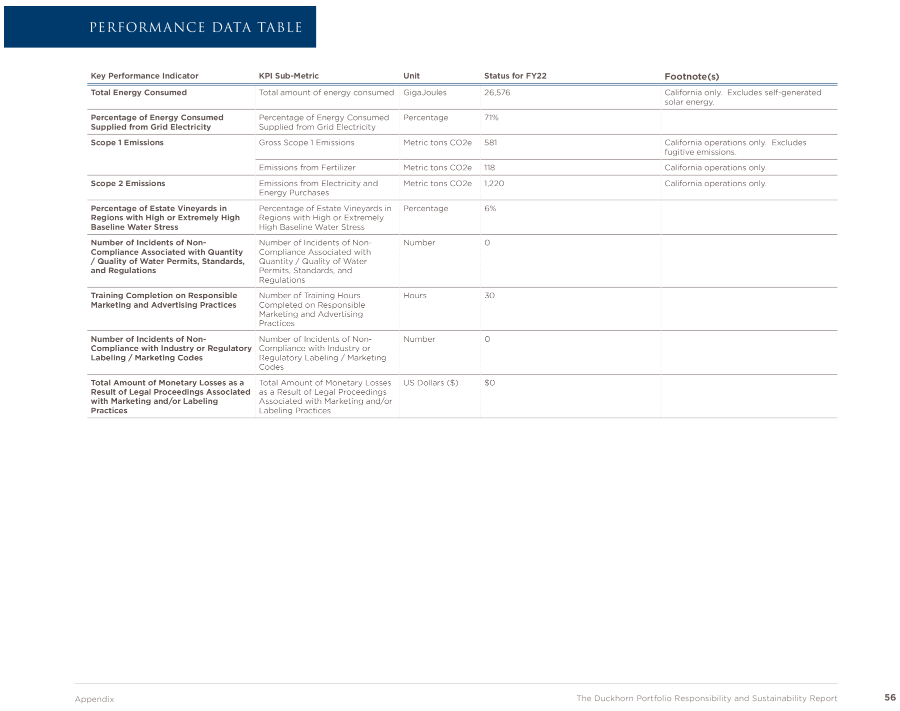
3.4 Output

In order to further process the extracted values, a consistent and machine-readable model output is crucial. To achieve this, the model will be prompted to respond with a JSON object using a specified schema. This output is enforced using llama.cpp's BNF grammar-based decoding and Pydantic type validation [8].

Listing 1 provides an example output extracted from the 2022 sustainability report by The Duckhorn Portfolio, Inc. [9]. The relevant page 56 of the report can be seen in Figure 1. The model outputs emission data using machine-readable numbers in a consistent unit. Furthermore, missing values are denoted using the `null` value. Additionally, the chunks used to arrive at the output are listed using their page number in the `sources` field. For interpretability purposes, a UI could convert these source ids into page numbers and display them to the

[8] `https://pydantic.dev`

[9] `https://www.duckhornportfolio.com/assets/client/File/BrandAssets/DWC/TDP_ESGReport-2022_FINAL.pdf`



PERFORMANCE DATA TABLE

Key Performance Indicator	KPI Sub-Metric	Unit	Status for FY22	Footnote(s)
Total Energy Consumed	Total amount of energy consumed	GigaJoules	26,576	California only. Excludes self-generated solar energy.
Percentage of Energy Consumed Supplied from Grid Electricity	Percentage of Energy Consumed Supplied from Grid Electricity	Percentage	71%	
Scope 1 Emissions	Gross Scope 1 Emissions	Metric tons CO2e	581	California operations only. Excludes fugitive emissions.
	Emissions from Fertilizer	Metric tons CO2e	118	California operations only.
Scope 2 Emissions	Emissions from Electricity and Energy Purchases	Metric tons CO2e	1,220	California operations only.
Percentage of Estate Vineyards in Regions with High or Extremely High Baseline Water Stress	Percentage of Estate Vineyards in Regions with High or Extremely High Baseline Water Stress	Percentage	6%	
Number of Incidents of Non-Compliance Associated with Quantity / Quality of Water Permits, Standards, and Regulations	Number of Incidents of Non-Compliance Associated with Quantity / Quality of Water Permits, Standards, and Regulations	Number	0	
Training Completion on Responsible Marketing and Advertising Practices	Number of Training Hours Completed on Responsible Marketing and Advertising Practices	Hours	30	
Number of Incidents of Non-Compliance with Industry or Regulatory Labeling / Marketing Codes	Number of Incidents of Non-Compliance with Industry or Regulatory Labeling / Marketing Codes	Number	0	
Total Amount of Monetary Losses as a Result of Legal Proceedings Associated with Marketing and/or Labeling Practices	Total Amount of Monetary Losses as a Result of Legal Proceedings Associated with Marketing and/or Labeling Practices	US Dollars ($)	$0	

Appendix The Duckhorn Portfolio Responsibility and Sustainability Report 56

Figure 1: Relevant emission page of the 2022 sustainability report by The Duckhorn Portfolio, Inc.

user. Note, however, that the source ids in the output are not guaranteed to actually contain the emission values. To alleviate this, a UI could display all pages provided to the model in the prompt and highlight those referenced by the model in the output. Importantly, this field is restricted to a maximum of five numbers, as excessively long source lists defeat the purpose of this information.

Listing 1: Example output.

```
{"scope_1":581,"scope_2":1220,"scope_3":null,"sources":[16,17,56,7]}
```

4 Evaluation

The selected models in Table 1 are evaluated using the manually-collected dataset of sustainability reports and associated emission values described in Section 2 and the developed system described in Section 3. The evaluation serves as benchmark for the task motivated in Section 1 as well as for the general use case of long-context structured information extraction.

4.1 Setup

The evaluation is run on a Linux system with an Nvidia GeForce RTX 2080 Ti GPU. Python 3.8.5 and CUDA 11.5.0 are used to run the experiments.

For the model comparison, all models are used in a consistent manner. The models are downloaded in the safetensors format[10] and converted to the GGUF format with float16 data type using llama.cpp scripts.

Each model is run using the prompt as described in Section 3.3. For models which were not trained using the ChatML format, the prompt is converted to the instruction format specified in their tokenizer configuration.

4.2 Results

Performance is measured using two metrics:

1. accuracy for every extracted emission value
2. source page retrieval accuracy

scope 1	scope 2	scope 3	avg of scopes	sources	model
49	34	54	**46**	53	mistral
33	31	56	**40**	48	openchat
12	8	5	**8**	3	qwen-1.8B
69	72	57	**66**	74	miqu
70	71	69	**69**	64	mixtral
65	62	69	**65**	77	lora

Table 3: Performance comparison of different models

Table 3 shows that Mixtral significantly outperforms the smaller models. Out of the smaller models, Mistral performs the best. Qwen-1.8B-Chat does not yield good results, but it is still surprising that it is even able to produce relevant outputs for emission values (which similar models such as phi-2 or stablelm-zephyr-3b failed to do). The Mistral model finetuned on Mixtral outputs on a different dataset remarkably nearly matches or partially exceeds Mixtral, which is important, as a 7B model is far easier to deploy than a 45B model.

It is notable that scope 2 accuracy is significantly lower for all models except Mixtral, likely due to oftentimes both market-based and location-based scope 2 emissions being present.

4.3 Numerical issues

Since language models struggle with numerical tasks, it is investigated whether wrong extractions are due to numerical errors instead of failures in logic. Table 4

[10] `https://huggingface.co/docs/safetensors/index`

shows the performance with accuracy measured by interpreting values within 10% of each other as matching. All models significantly perform better using this metric, confirming that they stuggle with reproducing numbers correctly. Notably, this is also true for Mixtral.

scope 1	scope 2	scope 3	avg of scopes	model
60	39	61	**53**	mistral
43	43	60	**49**	openchat
14	12	5	**10**	qwen-1.8B
78	75	71	**75**	mixtral
70	66	71	**69**	lora

Table 4: Performance comparison using a 10% margin for numerical accuracy

Furthermore, it is possible that extraction failures are due to wrong conversions of numbers into metric tons. This is investigated by computing accuracy irrespective of the unit. Again, the results in Table 5 show that the models perform better using this metric, confirming that they struggled with converting the numbers in the report to the correct unit.

scope 1	scope 2	scope 3	avg of scopes	model
54	36	59	**50**	mistral
35	35	62	**44**	openchat
17	11	8	**12**	qwen-1.8B
76	83	76	**78**	mixtral
79	78	81	**79**	lora

Table 5: Performance comparison using unit-agnostic accuracy

4.4 Prompt engineering

scope 1	scope 2	scope 3	avg of scopes	sources	model	type
38	31	44	**38**	52	mistral	no starting answer
49	34	54	**46**	53	mistral	starting answer

Table 6: Comparison of default ChatML prompts and ChatML prompts with the assistant's answer already started

Engineering good prompts is important to improve model performance. Table 6 compares default ChatML prompts and ChatML prompts with the assistant's answer already started by: `I have extracted the Scope 1, 2 and 3 emission values from the document, converted them into metric tons` `and put them into the following json object: '''json`. The results clearly show that the latter prompt leads to a better performance.

4.5 Self-extend

scope 1	scope 2	scope 3	avg of scopes	sources	failures	model	self-extend
40	32	50	**41**	30	40	openchat	no
33	31	56	**40**	48	4	openchat	yes

Table 7: Evaluation of the usefulness of self-extend

The openchat model is used to evaluate the usefulness of self-extend. Table 7 shows that running the openchat model without self-extend failed for 40 out of 100 reports, while it failed only on 4 reports when using self-extend. These crashes occur for long input prompts. This strongly shows that self-extend is able to significantly extend the context size of the model.

4.6 Finetuning

4.6.1 Overview

scope 1	scope 2	scope 3	avg of scopes	sources	learning rate	epochs	type
49	34	54	**46**	53	-	-	mistral
70	71	69	**69**	64	-	-	mixtral
65	62	69	**65**	77	2e-5	3	lora

Table 8: Performance comparison between base model and finetuned models using different techniques.

To investigate whether the performance of Mixtral can be achieved by a smaller, easier to deploy model, the Mistral-7B model is finetuned by training a LoRA on a dataset of 3233 sustainability reports and emission values extracted by Mixtral for 3 epochs. The results in Table 8 show that the finetuned model not only significantly outperforms the base model, but remarkably nearly matches and partially exceeds the performance of Mixtral, at a much smaller size.

4.6.2 DPO

The LoRA was also trained using DPO [8] by constructing a binary preferences dataset with randomly generated rejected outputs. Table 9 shows that the model finetuned using DPO performs significantly worse than the one using supervised finetuning.

scope 1	scope 2	scope 3	avg of scopes	sources	learning rate	epochs	type
49	34	54	**46**	53	-	-	base
43	37	55	**45**	60	5e-6	1	sft
32	32	49	**38**	58	5e-6	1	dpo

Table 9: Performance comparison between base model and LoRA models fine-tuned using supervised learning and DPO.

4.6.3 Rank comparison

scope 1	scope 2	scope 3	avg of scopes	sources	learning rate	epochs	rank
38	35	47	**40**	59	5e-6	1	4
43	37	55	**45**	60	5e-6	1	32
42	37	54	**44**	59	5e-6	1	64
56	38	57	**47**	56	5e-6	1	128

Table 10: Performance comparison between LoRA models with different ranks.

An important hyperparameter for training LoRA's is the LoRA rank. Table 10 shows that a rank of 128 performed the best, with 32 performing the second-best. However, this is not as clear anymore when a different learning rate is used, as can be seen in Table 11. Therefore, a rank of 32 was used for all other training runs.

scope 1	scope 2	scope 3	avg of scopes	sources	learning rate	epochs	rank
64	66	74	**68**	65	2e-5	1	32
60	58	69	**62**	69	2e-5	1	128

Table 11: Performance comparison between LoRA models with different ranks and learning rates.

4.6.4 Learning rate comparison

As mentioned above, a learning rate of 2e-5 led to better results than 5e-6 both at epoch 1 and 3, as shown in Table 12.

4.6.5 Epochs comparison

It can be seen above that performance increases significantly from epoch 1 to 3 for a learning rate of 5e-6. Table 13 shows the case for a learning rate of 2e-5. Interestingly, scope 2 and 3 accuracy decreases after the first epoch, while source accuracy increases significantly. At epoch 4, all metrics except scope 1 accuracy decrease. On average, the performance at 3 epochs is slightly better, therefore this model is used.

scope 1	scope 2	scope 3	avg of scopes	sources	learning rate	epochs	rank
43	37	55	**45**	60	5e-6	1	32
64	66	74	**68**	65	2e-5	1	32
60	57	68	**62**	67	5e-6	3	32
65	62	69	**65**	77	2e-5	3	32

Table 12: Performance comparison between LoRA models with different learning rates.

scope 1	scope 2	scope 3	avg of scopes	learning rate
64	66	74	**68**	2e-5
65	62	69	**65**	2e-5
67	59	68	**65**	2e-5

Table 13: Performance comparison between LoRA models with different learning rates and epochs.

4.6.6 Qwen-1.8B

As the Qwen-1.8B model is even easier to deploy than Mistral-7B, it was also investigated whether finetuning it increases its performance to a sufficient level. Qwen1.5-1.8B-Chat was used as base model due to its better support. Table 14 shows that the performance of the finetuned model is better, but still far from sufficient.

scope 1	scope 2	scope 3	avg of scopes	sources	learning rate	epochs	type
12	8	5	**8**	3	-	-	qwen-1.8B
3	5	22	**10**	6	2e-4	4	qwen-1.8B lora

Table 14: Performance comparison between the base Qwen-1.8B model and the finetuned Qwen-1.8B model.

4.7 Additional Findings

Interestingly, different self-extend hyperparameters yielded the optimal performance for different models. Table 15 lists the used values. The neighbour size especially is unusually and counterintuitively high.

5 Conclusion

Contributions of this project are:

- A manually-created evaluation dataset ($N = 100$),
- a synthetic finetuning dataset ($N = 3233$),

model	neighbour size	group size
mistral	8	1024
mixtral	8	1024
openchat	16	2048
qwen	64	2048

Table 15: Self-extend hyperparameters used in the evaluation.

- an evaluation of multiple models (1.8B-45B) covering prompt strategies, numerical issues and self extend,
- an evaluation of different finetuning configurations,
- a finetuned Mistral-7B model which nearly matches and partially exceeds Mixtral (45B) on this task, and
- a web demo (`https://huggingface.co/spaces/nopperl/emission-extractor`).

Additionally, this work provided general indications on using language models for long-context structured information extraction tasks.

5.1 Future Work

Noting the limitations of the presented work, there are numerous ways for future improvements.

In this work, all extracted chunks of the report are input into the model in a single run, relying on the capability of the model to resolve ambiguities and connect information from multiple pages. However, in most reports, the relevant values are contained in a single page, even if it is duplicated on multiple pages. In other words, there are no intra-page information dependencies. Hence, it would probably be possible to prompt each page individually, with a final prompt on all page extractions leading to the final output for the whole report. This would require a smaller context size, which would enable token-expensive prompting techniques such as chain-of-thought [11] or few-shot [2] prompting to be used. On the other hand, the system would have a longer runtime due to the necessity of running the model multiple times.

An important learning from collecting the dataset was that information is sometimes only displayed in graphs, which are not represented well in plain text form and require visual input to solve correctly. Multimodal (vision-language) models such as CogVLM [10] might perform better on these reports.

Similarly, while the focus was on large decoder-only language models, it may be interesting to test how smaller encoder(-decoder) models such as (XLM-)RoBERTa or DeBERTa perform. These models could be finetuned on the Mixtral output dataset. However, the average input sequence is far longer than their context size.

Furthermore, since the collected dataset also includes values for all scope 3 emission categories, these data could be extracted by a future system.

A Prompt

Listing 2: Simple instruction-following prompt.

```
You will be given chunks of text extracted from a
    sustainability report PDF document by a corporation.
    Each chunk is a single page of the PDF document and
    might not just contain flow text, but also text
    extracted from graphs, tables or similar structures.
    Your task is to extract the gross scope 1, 2 and 3
    greenhouse gas emissions. If values for multiple years
     are given, extract the value for the most recent year
    . For scope 2 emissions, extract only market-based
    emissions. Output this information as a JSON object in
     the <FORMAT>. Output only a single number for each of
     the scope 1, 2, and 3 emissions. If the chunks
    provide more detailed values, ignore them and output
    only the total value. If emission values are provided
    for subsidiaries, ignore them and only output the
    total value. It may be possible that the emission
    values are ambiguous (e.g. scope 1 and 2 emissions
    reported together as a single number) or don't exist
    at all. In this case, simply output 'null'. Be sure to
     only output a number if you are certain that it is
    correct. Extract scope 3 emissions based on generic
    emissions factors if multiple versions are given.
    Emission values may be reported in kilotons (thousand
    tons) or megatons (million tons). Convert these values
     into metric tons. The 'source' field should be a list
     of ids of the chunks which contain the extracted
    emission values.
<FORMAT>
{"scope_1": int | null, "scope_2": int | null, "scope_3":
     int | null, "sources": int[]}}
</FORMAT>
```

References

[1] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou,

Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[3] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023.

[4] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. LLM maybe longlm: Self-extend LLM context window without tuning. *CoRR*, abs/2401.01325, 2024.

[5] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[6] Xianming Li and Jing Li. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*, 2023.

[7] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[8] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[9] Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. Openchat: Advancing open-source language models with mixed-quality data. *CoRR*, abs/2309.11235, 2023.

[10] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. Cogvlm: Visual expert for pretrained language models. *CoRR*, abs/2311.03079, 2023.

[11] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.