Carbon Prompting: An Empirical Analysis of LLM-Based Requirement Classification

V Sathiyasuntharam, Kumkum Mishra, Devesh Kumar Gola, Lakshya Sharma
Department of CSE,Sharda University,Greater Noida,India
sathiya4196@gmail.com[1]
kumkummishra892004@gmail.com[2]deveshkumar8423gola@gmail.com[3]lakshaya4568@gmail.com[4]

Abstract - The growing use of Large Language Models (LLMs) in real-world applications has raised new questions about their environmental impact, especially during inference. While much research has focused on the energy demands of model training, this study draws attention to the often-overlooked emissions generated when these models are used at scale. We introduce the idea of carbon prompting, which examines how different prompt designs influence the energy use and carbon output of LLM inference. Using the PROMISE dataset for classifying functional and non-functional requirements, we tested nine prompting strategies with the LLaMA 3.2 model, including zero-shot, few-shot, Chain-of-Thought, detailed, JSON-based, self-critique, expert-engineer, instruction-few-shot, and no-explanation formats. Energy consumption and CO_2 emissions were tracked using CodeCarbon, allowing a detailed comparison of performance and environmental cost. The results show that simpler prompts-particularly no-explanation and zero-shot achieved the best balance between accuracy and energy efficiency. More elaborate prompts, although designed to elicit deeper reasoning, produced higher emissions without meaningful accuracy gains due to unnecessary token generation. These findings emphasize that prompt design should consider not only model performance but also sustainability, positioning environmental impact as a key metric in future AI development and evaluation.

Keywords - *Carbon emissions, Chain-of-Thought (CoT), CodeCarbon, Large Language Models (LLMs), prompt engineering, PROMISE dataset.*

I. Introduction

The growing concern over the sustainability and carbon cost of Large Language Model (LLM) inference has brought environmental efficiency to the forefront of artificial intelligence research. While training-related emissions have been widely studied, recent evidence indicates that inference the process of model usage at scale represents the dominant long-term contributor to energy consumption. This shift underscores the importance of examining how prompt design influences the environmental footprint of LLMs during real-world deployment.
Li et al. introduced Sprout, a framework that uses *generation directives* to cut inference emissions in models such as LLaMA 2, achieving over 40% reduction while maintaining output quality [1]. Similarly, Husom et al. developed MELODI, a monitoring and optimization framework that profiles energy consumption per prompt across various models and deployment environments, accompanied by an open dataset for energy-aware inference research [2]. Jegham et al. proposed an infrastructure-aware benchmarking methodology that measures energy, carbon, and water footprints across 30 LLMs using Data Envelopment Analysis (DEA) to evaluate eco-efficiency [3].
Prompt design has also been a focal point in Green AI research. Adamska et al. introduced Green Prompting, revealing that semantic structure and keyword choice often have a greater impact on energy use than prompt length [4]. Rubei et al. examined Prompt Engineering Techniques (PETs) for LLaMA 3 code generation tasks using the CodeXGLUE dataset, finding that customized prompt tagging significantly reduced inference energy without compromising accuracy [5]. Building on this, Niu et al. benchmarked different inference engines such as vLLM and DeepSpeed on H100 GPUs, breaking down total power consumption across setup, token generation, and hardware components [6].
Earlier foundational works established the principles of Green AI. Strubell et al. [7] advocated for transparent reporting of environmental metrics alongside accuracy results. Patterson et al. [8] highlighted that frequent inference in deployed systems could produce greater cumulative emissions than model

training, emphasizing the need for optimized inference pipelines. Lacoste et al. [9] introduced CodeCarbon, a widely used open-source tool that estimates CO_2 emissions based on hardware usage and regional grid intensity. Verdecchia et al. [10] later conducted a systematic review of Green AI literature, noting the imbalance between training-related studies and the limited attention to inference-stage impacts.
Taken together, these studies show that sustainability in AI cannot be achieved through performance optimization alone it also requires understanding prompt-level energy trade-offs. However, two clear research gaps remain:

1. Existing frameworks such as Sprout [1], MELODI [2], and CodeCarbon [9] primarily focus on model-level or benchmark-level energy profiling rather than task-specific prompt optimization.
2. Prior empirical works such as Green Prompting [4] and Rubei et al. [5] have explored energy impacts in general NLP and code generation but have not investigated software engineering contexts like requirements classification.

To address these gaps, this study presents an empirical evaluation of prompt engineering strategies in LLM-based software requirement classification, using the PROMISE dataset. By combining performance metrics with real-time energy and carbon tracking, this research aims to quantify how prompt complexity influences both model effectiveness and environmental sustainability.

II. Literature Review

The sustainability and carbon footprint of Large Language Model (LLM) inference—particularly in relation to prompt design have become a central theme in contemporary AI research. In [1], the Sprout framework was introduced, utilizing *generation directives* to minimize the carbon footprint of LLM inference (e.g., LLaMA 2) by over 40% while preserving output quality. Similarly, a framework titled MELODI was presented in [2], enabling detailed energy profiling at the prompt level across various LLMs and deployment environments, along with the release of an accompanying open dataset for continued research. An infrastructure-aware benchmarking methodology was proposed in [3] to quantify energy, carbon, and water footprints during inference across 30 LLMs. By applying Data Envelopment Analysis (DEA), the study assessed eco-efficiency based on response length and query volume. A comprehensive empirical study named Green Prompting [4] analyzed how prompt characteristics such as length, semantics, and keyword structure—affect energy consumption. The findings indicated that *semantic intent* often has a greater impact on energy usage than prompt length alone.
In [5], the authors examined Prompt Engineering Techniques (PETs) and their influence on LLaMA 3's energy use during code generation tasks. They found that structured tagging and domain-specific templates significantly reduced inference energy while maintaining performance accuracy. The benchmarking study presented in [6] further decomposed inference into setup and token generation phases, comparing energy use across GPUs, CPUs, and DRAMs in inference engines such as vLLM and DeepSpeed.
The foundations of Green AI were established in [7], emphasizing the need to include environmental cost reporting alongside conventional performance metrics. Later, Patterson et al. [8] highlighted that repeated inference during real-world deployment can surpass training emissions, calling for more efficient inference operations. The CodeCarbon tool introduced in [9] became a standard for estimating CO_2 emissions by tracking hardware energy usage and regional carbon intensity. Lastly, a systematic review conducted in [10] provided a broader overview of sustainable AI research, observing that while training-related emissions are well studied, inference-level sustainability remains largely underexplored.
Research within the domain of Green AI has consistently emphasized the importance of transparently reporting energy and carbon metrics alongside conventional performance measures [7], [10]. Tools such as CodeCarbon have made such evaluations practical by integrating hardware-level monitoring with regional carbon intensity data [9]. Building on this foundation, Patterson et al. [8] highlighted that emissions during inference particularly in high-frequency deployment scenarios can surpass those from training, underscoring the urgency of optimizing inference efficiency.
In response to these concerns, recent developments such as Sprout [1] and MELODI [2] have introduced prompt-level tracking and optimization frameworks, quantifying the carbon footprint associated with

varying prompt characteristics. Complementary empirical efforts, including Green Prompting [4] and infrastructure-level benchmarking [3], have offered valuable insights into how prompt semantics, model scale, and inference frequency influence overall energy consumption. Furthermore, Rubei et al. [5] demonstrated that structured prompt engineering techniques such as tagging and controlled instructions-can substantially lower the energy consumption of LLaMA 3 during code generation without compromising accuracy.
Despite these advancements, prompt-level energy trade-offs in software engineering contexts remain underexplored. Specifically, there is a lack of systematic research addressing how different prompting strategies affect both performance and environmental cost in tasks like software requirement classification. The present study addresses this gap by empirically evaluating the carbon impact of diverse prompting strategies using the PROMISE dataset, offering novel empirical insights into the intersection of prompt engineering, efficiency, and sustainability.

Table I. *Literature Summary*

Year, Author & Title	**Focus Area**	**Algorithm / Method Used**	**Research Gap**
2024, Li et al., *Sprout: Carbon-Efficient Generation Directives*	Reduce inference emissions	Rule-based generation directives	Limited to text-generation; not applied to requirement classification
2024, Husom et al., *MELODI: Energy Profiling Framework*	Prompt-energy monitoring	Profiling with monitoring tool	Focused on profiling; lacks optimization for SE tasks
2025, Jegham et al., *Benchmarking Eco-Efficiency of LLMs*	Multi-model benchmarking (30 LLMs)	Data Envelopment Analysis (DEA)	No prompt-level or SE-specific analysis
2025, Adamska et al., *Green Prompting Study*	Impact of prompt semantics vs. length	Empirical analysis across tasks	Did not quantify requirement classification prompt trade-offs
2025, Rubei et al., *Prompt Engineering for LLaMA-3*	Energy in code generation tasks	Custom tagging in PETs	Limited to code generation; no SE requirement focus
2025, Niu et al., *LLM Inference Engine Benchmark*	System-level energy breakdown	GPU/CPU/DRAM profiling	Hardware-centric; ignores prompt design
2019, Strubell et al., *Green AI*	Transparency in AI research	Reporting framework	No prompt-level methodology proposed
2021, Patterson et al., *Carbon Cost of Inference*	Inference vs. training emissions	Emission modeling & measurement	Did not address prompt efficiency
2020, Lacoste et al., *CodeCarbon Toolkit*	Carbon tracking library	Estimation via hardware + geography	Tracking only; not prompt optimization
2023, Verdecchia et al., *Systematic Green AI Review*	Literature review	Review methodology	Training-centric; neglects inference-prompt interplay

III. METHODOLOGY

This study systematically explores how different prompt engineering strategies influence both the performance and environmental footprint of Large Language Model (LLM)–based software requirement classification. The methodology was designed to be robust, reproducible, and data-driven, combining local inference with real-time energy and carbon monitoring. The overall framework enables direct empirical comparison of diverse prompt configurations in terms of accuracy, precision, and energy efficiency.

A. Methodological Framework and Supporting Infrastructure

All experiments were executed locally under controlled conditions to ensure precision and reproducibility of measurements.

- **LLM Deployment:** The *LLaMA3-8B* model was deployed locally using the **Ollama** framework, allowing on-device inference and fine-grained monitoring of computational resources.
- **Energy Tracking:** The **CodeCarbon** library was integrated to measure energy consumption (in Joules) and estimate CO_2 emissions (in gCO_2eq) based on regional electricity carbon intensity.
- **Dataset:** Experiments used the **PROMISE** dataset containing 655 annotated software requirements categorized as Functional and Non-Functional. The dataset was split into training and testing subsets using stratified sampling.

Enhanced Experimental Control:

a) A **global CodeCarbon tracker** was initialized once per experimental run to minimize measurement noise and improve temporal accuracy.

b) **Retry and timeout mechanisms** were added to the API communication layer to handle prolonged or unstable inference sessions.

c) **A CPU warm-up routine** was executed prior to each run to stabilize processor temperature and clock speed.

B. Prompt Engineering Methodology

Nine distinct prompting strategies were evaluated to assess their effect on both performance and environmental metrics:

1. **Zero-Shot:** Direct task instruction without examples.
2. **Few-Shot:** Includes one to three example pairs to provide context.
3. **Chain-of-Thought (CoT):** Encourages reasoning through intermediate steps before output.
4. **Instruction Few-Shot:** Hybrid format combining task definitions and sample inputs.
5. **Detailed Prompting:** Elaborate, context-rich task instructions.
6. **Expert Engineer:** Domain-specific phrasing emulating software engineering terminology.
7. **JSON Output Prompt:** Prompts structured to elicit machine-readable JSON outputs.
8. **No-Explanation:** Prompts restricting responses strictly to classification labels.
9. **Self-Critique:** Prompts instructing the model to generate an answer, critique it, and refine the result.

For each category, **30 prompt variants** were generated, varying along two dimensions:

- **Length:** Short (<500 characters), Medium (501–3000), Long (>3000).
- **Complexity:** Low (simple syntax), Medium (moderate domain terms), High (technical phrasing).

Each variant was automatically tagged with metadata - *Prompt_Type*, *Length_Type*, *Prompt_Complexity*, and *Token_Length* - using a tokenizer for later analysis.

C. Execution Protocol

The overall experimental process, illustrated in Fig. 1, followed a carefully structured pipeline:

1. **System Warm-Up:** A CPU-intensive Fibonacci routine was run before each session to stabilize power draw.
2. **Repeated Execution:** Each prompt configuration was executed **30 times** to ensure statistical reliability.
3. **Rest Intervals:** A **60-second pause** between configurations allowed the system to return to baseline.
4. **Stratified Sampling:** Dataset partitioning preserved the same functional/non-functional class ratio across repetitions.

Algorithm 1: *Carbon Prompting Evaluation Framework*

```
Input: Dataset D, Prompt Variants P, LLM Model M
Output: Performance & Environmental Metrics
1. Load Dataset D and preprocess
2. For each prompt p in P:
    a. Warm-up system
    b. For i = 1 to 30:
        i. Split D into train/test (70:30, stratified)
        ii. Measure start time, start CodeCarbon tracker
        iii. Run inference using M with prompt p
        iv. Stop timer and tracker
        v. Record Accuracy, Precision, Recall, F1,Energy, Carbon Emissions
3. Store results in structured dataset
4. Apply statistical analysis (Friedman + Nemenyi tests)
```

Algorithm 2: Prompt Efficiency Score (PES) Computation

```
Input: Results dataset R with Accuracy, Energy, Carbon Emissions per prompt
Output: Prompt Efficiency Score (PES) per prompt variant
1. For each record r in R:
    a. Extract Accuracy (Acc), Energy (E), Carbon Emissions (C).
    b. Compute PES = Acc / (E + λ·C)
        i. where λ is a scaling factor to normalize emissions into energy units.
2. Store PES alongside existing metrics.
3. Rank prompts by PES to identify most sustainable high-performance prompts.
4. Output best-performing prompt(s) based on PES ranking.
```

D. Metrics and Analysis

1) Performance Metrics:

$$Accuracy = \frac{Total\ Number\ of\ Predictions}{Number\ of\ Correct\ Predictions}$$

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2) Efficiency Metrics:

Energy Consumption: $\text{Energy (J)} = P_{\text{hardware}} \cdot T_{\text{inference}}$

$$\boldsymbol{Carbon\ Emissions\ (gCO_2eq)} = Energy\ (kWh) \times Carbon\ Intensity(\frac{gCO_2eq}{kWh})$$

$$\boldsymbol{Inference\ Time\ (s)} = t_{end} - t_{start}$$

3) Composite Metric (Prompt Efficiency Score, PES):

Performance per Energy Sustainability (PES): $\text{PES} = \frac{\text{Energy}}{\text{Accuracy}}$

This measures sustainability-adjusted performance, allowing direct comparison of prompts.

E. Statistical Methods

Given repeated measures across prompt types, **non-parametric tests** were applied:

- **Friedman Test** — detects significant differences in median performance and efficiency metrics.
- **Nemenyi Post-Hoc Test** — identifies specific prompt pairs with significant differences.
- Significance threshold: p<0.05p < 0.05p<0.05.

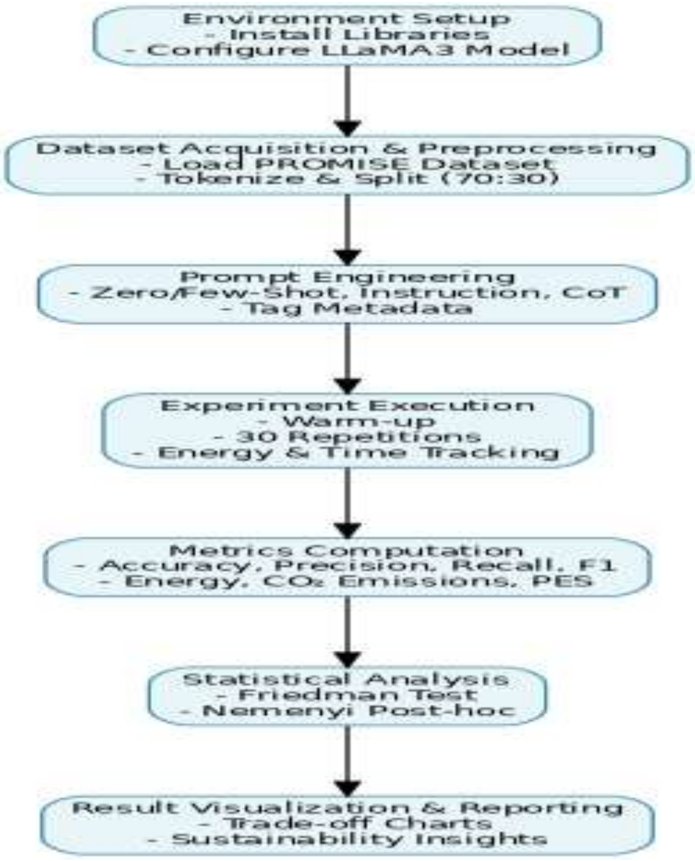

Fig 1: Workflow of the Carbon Prompting Evaluation Framework

IV. Experimental Setup

Experimental work for *Carbon Prompting* was carried out in a controlled environment to ensure reproducibility and precision in both performance and energy evaluation. All experiments were executed locally on a MacBook Pro equipped with an Intel i7 processor and 16 GB RAM, with inference performed on the CPU through the Ollama framework. This configuration minimized variability introduced by cloud-based execution and provided stable, measurable energy readings.

The CodeCarbon library was employed to monitor power consumption and CO_2 emissions, while Python libraries such as *Pandas*, *NumPy*, *Matplotlib*, and *Scikit-learn* were used for preprocessing, statistical analysis,and visualization.

A uniform execution protocol was followed across all experiments. A CPU warm-up was performed prior to each run to stabilize temperature and power draw, thereby reducing hardware-induced measurement bias. Each prompt configuration was executed 30 times to establish statistical reliability. The dataset was split using stratified sampling with a 70:30 train–test ratio, preserving the balance between Functional and Non-Functional requirement classes. To eliminate residual system load between runs, a 60-second rest interval was imposed after each configuration.

The entire procedure was fully automated via a Python-based execution pipeline that integrated CodeCarbon and Ollama. The emissions tracker was initialized once at the beginning of each experiment and stopped at the end, with total emissions proportionally allocated across individual inference runs according to their duration. This design improved both accuracy and computational efficiency of the measurements.

The pipeline received raw requirement statements as input, which were transformed into nine prompt variants- *zero_shot, few_shot, cot, detailed, instruction_few_shot, json_output, expert_engineer, no_explanation,* and *self_critique*. Each prompt was automatically labeled with metadata including *prompt type, token length,* and *complexity level*.
The output consisted of a structured CSV log documenting every experimental iteration. Each record contained detailed metadata: dataset name, iteration number, model identifier, prompt type, token length, accuracy, precision, recall, F1-score, inference time, energy consumed, and carbon emitted.
This schema ensured full traceability and enabled integrated analysis of both model performance and environmental footprint.

Table II. A sample record from the final dataset is illustrated below:

Attribute	**Value**
Dataset Name	req_dataset
Iteration Number	12
LLM Name	llama3.2
Prompt Type	zero_shot
Token Length	45
Accuracy	0.25
Precision	0.27
Recall	0.23
F1-Score	0.24
Inference Time (s)	20.3
Energy (J)	0.135
Carbon Emission (gCO2eq)	0.000135

This setup not only enabled reproducibility but also ensured that every experiment could be mapped precisely to its configuration. The rigorous documentation of both prediction results and energy usage allowed for a joint evaluation of model performance and sustainability.

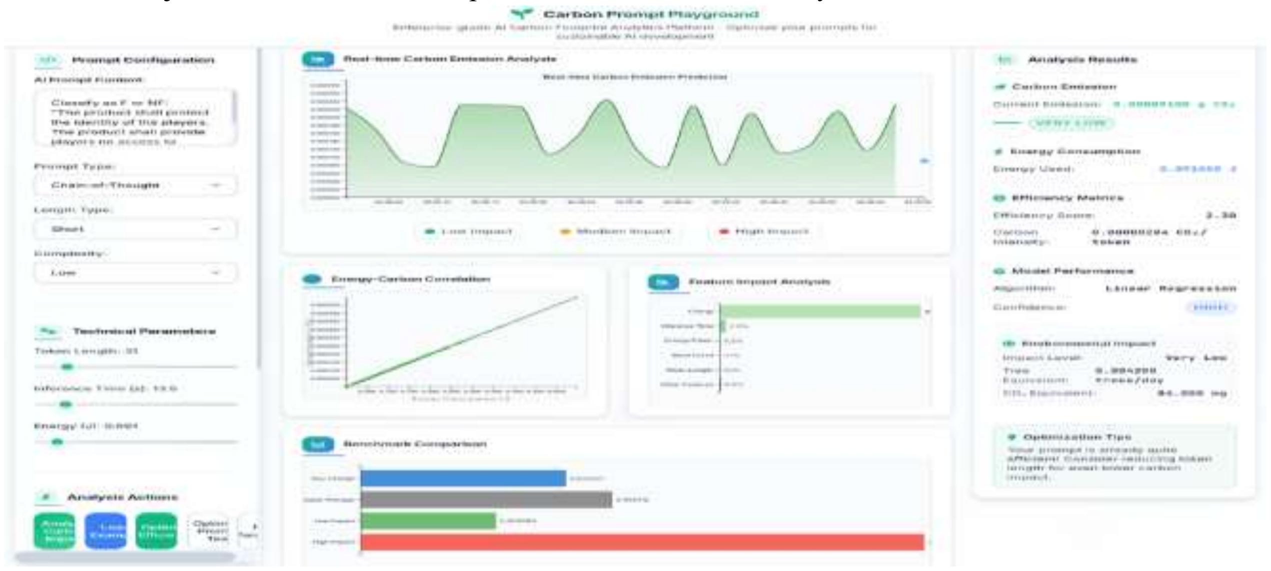

Fig. 3 illustrates the integrated visualization dashboard, combining four perspectives—accuracy, inference time, energy consumption, and carbon emissions—offering a comprehensive view of prompt-level trade-offs.

To support visualization, several perspectives were used to highlight trade-offs across prompt types. Bar charts compared mean classification accuracy, showing that the no_explanation and zero_shot prompts achieved the highest accuracy and consistency. Scatter plots of correctness versus energy consumption revealed that cot and detailed prompts consumed higher energy while yielding lower accuracy. Box plots of inference time and carbon emissions further exposed that complex prompts exhibited greater variance and higher mean values.

Table III. Summary of Performance & Environmental Metrics Across Prompt Types

Prompt Type	**Mean Accuracy (%)**	**Mean Inference Time (s)**	**Mean Energy (J)**	**Mean Carbon Emissions (gCO_2eq)**
cot	8.3	25.5	0.172	0.000172
detailed	27.8	45.1	0.304	0.000304
expert_engineer	13.9	20.1	0.135	0.000135
few_shot	19.4	18.4	0.124	0.000124
instruction_few_shot	13.9	19.8	0.128	0.000128
json_output	33.3	20.3	0.135	0.000135
no_explanation	44.4	17.6	0.112	0.000112
self_critique	30.6	21.4	0.142	0.000142
zero_shot	30.6	20.0	0.135	0.000135

This unified presentation clearly demonstrates that prompt design directly impacts both performance and sustainability. Complex prompt types (*cot, detailed*) consume more energy and time without proportional gains in accuracy, whereas concise prompting strategies (*no_explanation, zero_shot*) achieve an optimal balance between classification accuracy, inference efficiency, and carbon footprint reduction.

Visualisation & Results

To gain insights into the performance and sustainability trade-offs of different prompting strategies, the collected results were analysed and visualized through multiple plots. These visualizations highlight inference efficiency, carbon impact, correctness, and accuracy distribution across prompt types.

Inference Time Distribution:

Figure 4 presents a boxplot of inference time across nine prompting strategies: *zero_shot, few_shot, chain_of_thought (CoT), detailed, instruction_few_shot, json_output, expert_engineer, self_critique,* and *no_explanation.*

The results show that detailed prompting incurs the highest median inference time and widest variability, followed by CoT, both of which display significant outliers indicating unstable performance. In contrast, no_explanation and zero_shot prompts demonstrate lower and more consistent inference times, indicating higher computational efficiency and stability.

Carbon Emission by Prompt Type:

Figure 5 illustrates the carbon emissions (gCO_2eq) associated with each prompt type. Although the absolute emission values remain low due to localized CPU inference, a noticeable trend appears: detailed and CoT prompts produce higher emissions compared to zero_shot and no_explanation. This increase in emissions directly correlates with longer token generation and reasoning verbosity, confirming that more complex prompt formulations lead to higher Environ mental costs.

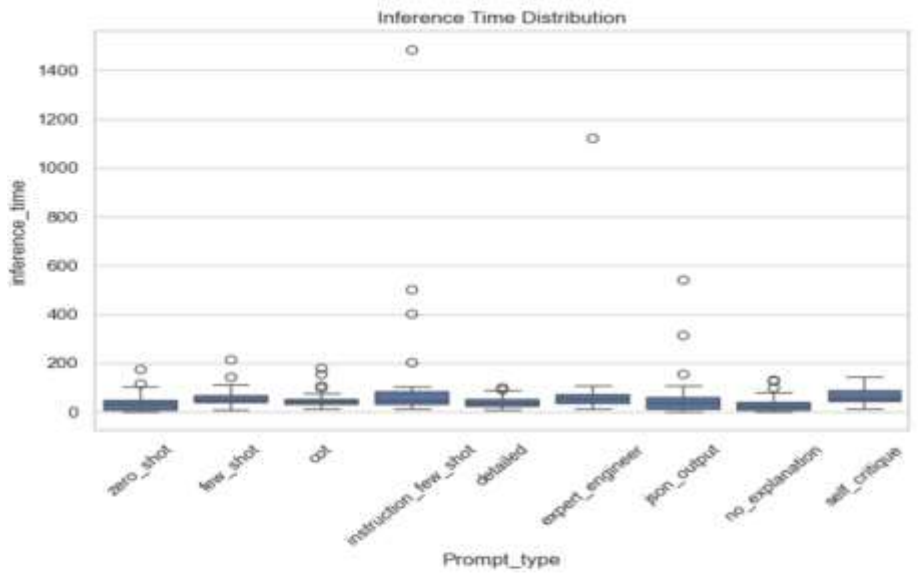

Fig 4: Showing the plot of inference time taking vs prompt type

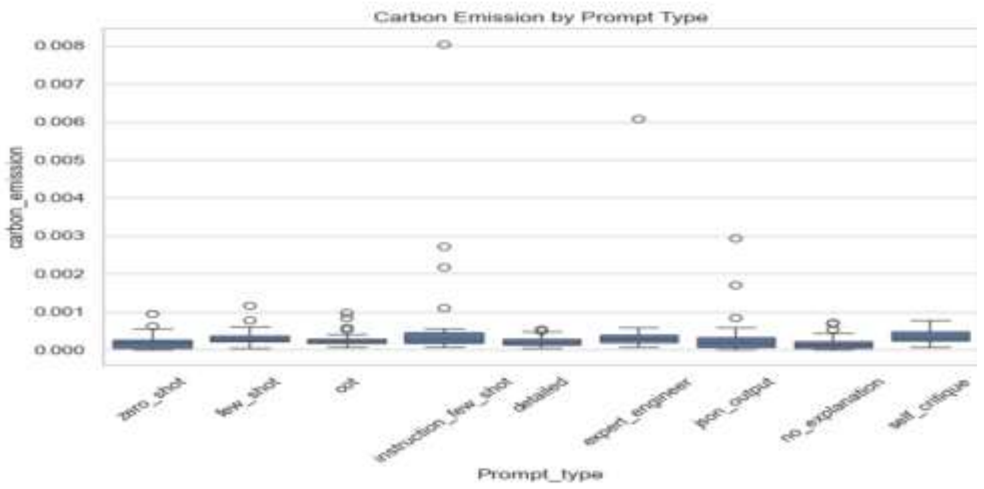

Fig 5: *Showing the plot carbon emission vs prompt type*

Accuracy by Prompt Type:
Figure 7 provides a comparative bar plot of mean accuracy across all prompt categories. Among the evaluated strategies, no_explanation achieved the highest mean accuracy (~0.44), followed by json_output (~0.33) and zero_shot (~0.31). In contrast, CoT and instruction_few_shot prompts showed the lowest performance, underlining the inefficiency of overly complex prompt structures in this classification task

Energy vs. Correctness:
Figure 6 visualizes the relationship between energy consumption and correctness across all prompt runs. The majority of data points cluster near lower energy consumption levels, with correctness values distributed between 0 and 1. The absence of a clear positive correlation between higher energy use and improved correctness indicates that greater computational effort does not necessarily yield better accuracy. This reinforces that simpler prompt types, despite lower energy demands, can achieve competitive or superior correctness.

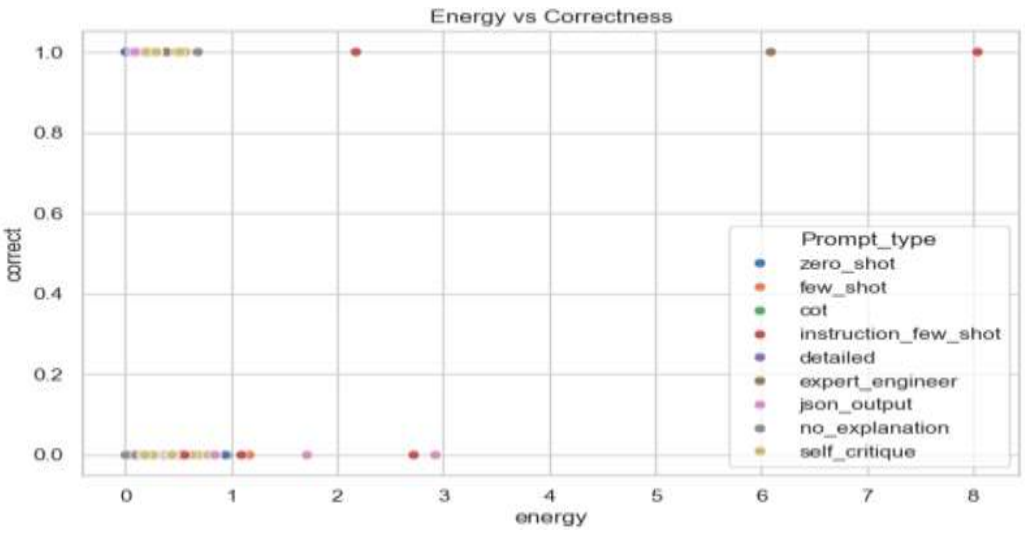

Fig 6: Showing the plot Energy vs Correctness

.

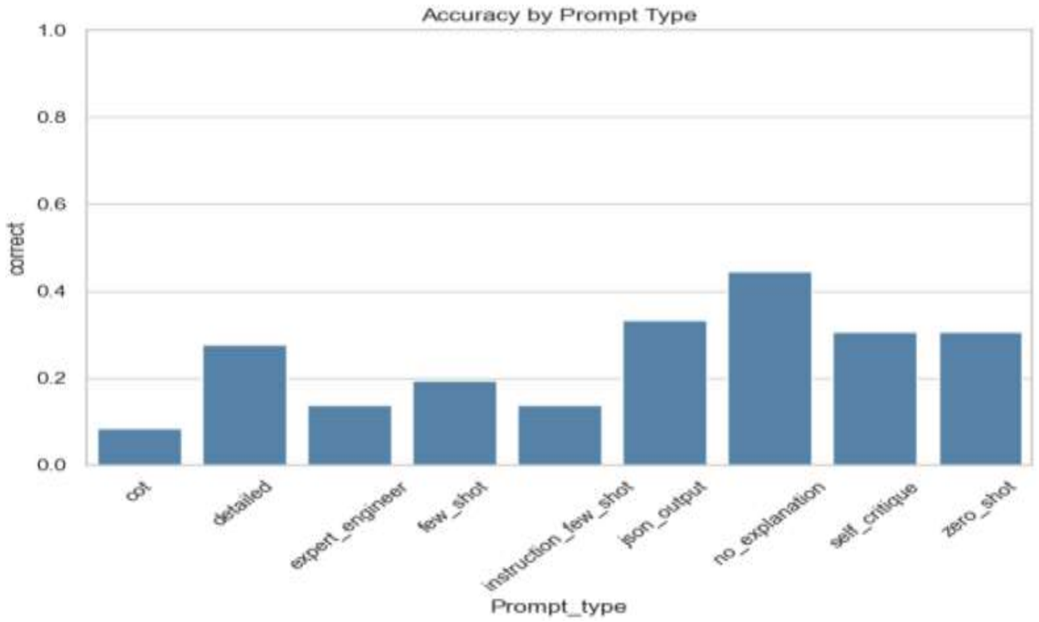

Fig 7: Showing the plot of accuracy by prompt type

These results collectively establish that **no_explanation** and **zero_shot** prompts offer the most favorable balance between accuracy, inference time, and sustainability. Conversely, complex strategies such as **CoT** and **detailed** prompts result in excessive reasoning overhead, longer processing times, and higher carbon emissions—without contributing to proportional accuracy gains. This demonstrates that simplicity in prompt design leads not only to better efficiency but also to a measurable reduction in environmental footprint.

V. Discussion & Future Enhancement

The findings of the current study confirm a recurring theme: raising prompt complexity isn't always associated with better performance and in fact, frequently results in the opposite when faced with binary classification tasks. The empirical results point out an unmistakable negative return on complexity, whereby

techniques like Chain-of-Thought (CoT), verbose, and instruction-based prompts significantly boosted computational time and energy consumption yet resulted in lower classification accuracy. This contrasts with previous NLP work on reasoning-demanding areas, where CoT generally improves performance [1].
The results prove that prompt engineering's effectiveness is very task-specific. Requirement classification itself is a straightforward binary task by nature. The LLaMA3.2 model, trained on the massive datasets, contains adequate latent knowledge to excel with the short and direct instructions like zero-shot or no_explanation prompts. Increasing layers of reasoning or verbose instructions leads to "over-reasoning," resulting in overly lengthy outputs that compromise precision and use more energy through the boosted token generation.
According to these findings, the paper presents and encourages the notion of Green Prompting, focusing on energy-efficient and task-specific prompt design. The following guiding principles are put forth for AI sustainable development:

1) Adopt the Principle of Parsimony: Begin with the most straightforward effective prompt (e.g., zero-shot or no_explanation). Escalate complexity only if it brings quantifiable accuracy gains. Be concise, as longer outputs are a direct factor in token generation and consequently energy consumption [6].
2) Practice Task-Appropriate Prompting: Methods such as CoT and elaborate reasoning should be left for multi-step logical inference tasks. For easier binary tasks, such methods are computationally inefficient and environmentally wasteful.
3) Enforce Output Constraints: The prompts must strictly limit output format (e.g., "Only give F or NF"). This minimizes verbosity, guarantees consistent parsing, and prevents unnecessary token emissions, which results in improved energy efficiency.

Looking ahead, several future research directions emerge to strengthen this field and enhance sustainable AI evaluation frameworks:

1. **Development of a Prompt Efficiency Score (PES):**
   Introduce a unified metric combining accuracy and sustainability such as

$$PES = \frac{Accuracy}{Energy + \lambda \times CarbonEmission}$$

This measure would enable authors to compare prompt designs on both performance and carbon footprint, thereby encouraging the use of green models.

2. Extended Empirical Studies: Expand experimentation to several open-source LLMs (e.g., Mistral, Falcon, Gemma) and diverse datasets to confirm if the "negative return on complexity" principle generalizes across domains other than requirement classification [1].
3. Automated Sustainable Prompt Optimization: Future systems would be able to utilize automated prompt optimization frameworks that optimize for Pareto-optimal trade-offs—balancing maximum accuracy against minimum energy consumption and emissions via adaptive prompt tuning.

VI. Conclusion

The mass deployment of Large Language Models (LLMs) has brought their own environmental impact-namely, inference costs-to the center of AI sustainability debates. This paper presents and empirically confirms the idea of "carbon prompting," demonstrating that prompt design has a large impact on an LLM query's energy consumption and carbon footprint. With a controlled experiment on the classification of software requirements, the results show that there is no one "best" prompt that works everywhere; short, specific prompts were not just less costly in terms of energy but even equivalent in performance to long ones. The results also show that calling upon sophisticated reasoning for mundane tasks of classification is computationally wasteful and hurts performance. Prompt design becomes a prime factor, and not a secondary issue, with sustainable AI. Frugal, task-specified prompt engineering is therefore a fundamental principle of AI development responsibly vital as the community strives toward more powerful but ecologically aware models.

REFERENCES

[1] Özcan, M. (2025). *Quantifying the energy consumption and carbon implications of LLM inference.* arXiv. https://arxiv.org/abs/2507.11417

[2] Cutter Consortium. (2023, November 15). *Large language models: What's the environmental impact?* Cutter Consortium. https://www.cutter.com/article/large-language-models-whats-environmental-impact

[3] Bailey, N. (2024, September 2). *The carbon footprint of LLMs: A disaster in waiting?* Medium. https://nathanbaileyw.medium.com/the-carbon-footprint-of-llms-a-disaster-in-waiting-6fc666235cd0

[4] Faiz, A., Kaneda, S., Wang, R., Osi, R., Sharma, P., Chen, F., & Jiang, L. (2023). *LLMCarbon: Modeling the end-to-end carbon footprint of large language models.* arXiv. https://arxiv.org/abs/2309.14393

[5] Rubei, R., Moussaid, A., di Sipio, C., & di Ruscio, D. (2025). *Prompt engineering and its implications on the energy consumption of large language models.* arXiv. https://arxiv.org/abs/2501.05899

[6] Prompting Guide. (n.d.). *Research papers.* PromptingGuide.ai. https://www.promptingguide.ai/papers

[7] Williams, A. (2025, June 16). *Smarter prompts for a more sustainable future?* Blog@CACM, Communications of the ACM. https://cacm.acm.org/blogcacm/smarter-prompts-for-a-more-sustainable-future.

[8] Google Cloud. (n.d.). *What is prompt engineering?* Google Cloud - Discover. https://cloud.google.com/discover/what-is-prompt-engineering.

[9] Federiakin, D., Molerov, D., Zlatkin-Troitschanskaia, O., & Maur, A. (2024). *Prompt engineering as a new 21st-century skill. Frontiers in Education.* https://www.frontiersin.org/articles/10.3389/feduc.2024.1366434/full.

[10] *Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv. https://arxiv.org/abs/2402.07927*.

[11] Sustainability-Directory. (2025). *Green AI principles.* Sustainability-Directory. https://climate.sustainability-directory.com/term/green-ai-principles.

[12] *Deutsche Telekom. (2024). Principles for green artificial intelligence. Deutsche Telekom. https://www.telekom.com/en/company/details/principles-for-green-artificial-intelligence-1077104*.

[13] Green Software Foundation. (2025, May 13). *Green AI position paper.* Green Software Foundation. https://greensoftware.foundation/articles/green-ai-position-pape.

[14] VerifyWise. (n.d.). *Green AI principles.* VerifyWise AI Lexicon. https://verifywise.ai/lexicon/green-ai-principles.

[15] Rubei, R., Moussaid, A., di Sipio, C., & di Ruscio, D. (2025). *Prompt engineering and its implications on the energy consumption of large language models* [ResearchGate mirror]. ResearchGate. https://www.researchgate.net/publication/387953996_Prompt_engineering_and_its_implications_on_the_energy_consumption_of_Large_Language_Models.

[16] Dauner, M., & Socher, G. (2025). *Energy costs of communicating with AI. Frontiers in Communication, 10*, Article 1572947. https://doi.org/10.3389/fcomm.2025.1572947.

[17] Strubell, E., Ganesh, A., & McCallum, A. (2019). *Energy and Policy Considerations for Deep Learning in NLP*. arXiv. https://arxiv.org/abs/1906.02243

[18] Patterson, D., Gonzalez, J. E., Le, Q., Dean, J., & others (2021). *Carbon Emissions and Large Neural Network Training*. arXiv. https://arxiv.org/abs/2104.10350

[19] Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). *Green AI.* Communications of the ACM. https://dl.acm.org/doi/10.1145/3381831