Keyword Matters: Unveiling the Energy Sensitivity of On-Device LLM Prompting

Ruiyi Tao
Juanita High School
Redmond, Washington, USA
taoroy960@gmail.com

Xiaolong Tu
Georgia State University
Atlanta, Georgia, USA
xtu1@gsu.edu

Haoxin Wang
Georgia State University
Atlanta, Georgia, USA
haoxinwang@gsu.edu

***Abstract*—Large Language Models (LLMs) are increasingly deployed on mobile and embedded devices to improve privacy and reduce network latency. Yet on-device inference faces a fundamental constraint: high energy consumption on battery-powered, resource-limited hardware. While model compression and runtime acceleration have been widely studied, the effect of*prompt design* on energy efficiency remains underexplored. This paper presents an empirical study of the relationship between prompt wording and energy consumption for on-device LLMs. Using real power measurements collected on a smartphone, we quantify how linguistic features, particularly imperative keywords and instruction structure, affect decoding length and total energy. Our results show consistent energy differences across verbs and tasks, indicating that prompt engineering is a lightweight lever for improving energy efficiency.**

I. Introduction

Large Language Models (LLMs), such as GPT-4, LLaMA 3, and DeepSeek, have demonstrated remarkable capabilities in text generation, reasoning, and problem solving. They now power applications ranging from virtual assistants and coding copilots to educational and creative tools. To support these workloads, most commercial LLMs are currently deployed in cloud environments, where large-scale computing clusters provide sufficient memory, processing, and cooling resources to handle billions of parameters and real-time user requests [1], [2].

However, cloud-based inference introduces growing concerns about data privacy, latency, and energy cost. Sensitive user data must travel across networks to remote servers, creating potential privacy risks and increasing response delays. As a result, researchers and industry developers have begun exploring on-device LLM deployment, running compact models directly on mobile phones, embedded systems, and IoT devices [3], [4], [5], [6]. This emerging trend offers improved privacy and offline availability, but also exposes a fundamental constraint: limited energy and computing resources.

Unlike data centers with continuous power supply and active cooling, edge devices are battery-powered and thermally constrained. The high computational demand of autoregressive token generation leads to substantial energy draw on CPUs, GPUs, or NPUs, shortening device battery life and limiting sustained performance [7], [8]. As model inference scales with prompt length, decoding steps, and reasoning depth, even small inefficiencies in model execution or input design can result in significant energy overhead. Therefore, understanding and reducing energy consumption has become a crucial step toward enabling practical and sustainable on-device intelligence.

While prior research has optimized model architectures through quantization, pruning, and distillation, the effect of prompt design on energy consumption remains underexplored. Recent works suggest that different prompt wordings can trigger distinct reasoning paths, token lengths, and attention activations, potentially altering power usages. This raises an intriguing question: *Can prompt engineering serve as a lightweight mechanism for energy-aware control during on-device LLM inference?*

In this work, (i) We present a controlled measurement study of keyword-level prompt effects on decoding energy. (ii) We define a measurement protocol and metrics (TTFT, TBT, energy split into prefill/decoding) and report cross-model, cross-task trends. We analyze how different linguistic patterns, especially keywords, affect runtime behavior and power draw. Our findings highlight that prompt formulation is not only a semantic or stylistic factor, but also an energy-relevant one.

II. Related Work

A. LLM Optimization and On-Device Deployment

Recent advances in system and algorithm co-design have significantly reduced LLM inference cost. FlashAttention reorders attention computation to minimize memory access and achieve GPU speedups [9], while vLLM introduces PagedAttention to mitigate KV-cache fragmentation and improve long-context throughput [10]. Speculative decoding accelerates generation through draft-and-verify prediction [11]. Model-side compression complements these efforts. For example, QLoRA enables memory-efficient fine-tuning on low-bit models [12].

Beyond cloud servers, compact transformer architectures such as MobileLLM [13] demonstrate the feasibility of sub-billion-parameter LLMs on edge devices. Frameworks like MLC-LLM and TensorRT-LLM further optimize execution pipelines for mobile and embedded NPUs. Meanwhile, sustainability studies such as CarbonTracker [14] and the BLOOM carbon report [15] provide tools for quantifying energy and emissions in AI training and inference. However, most prior work focuses on hardware or model efficiency rather than user-driven linguistic effects on energy consumption.

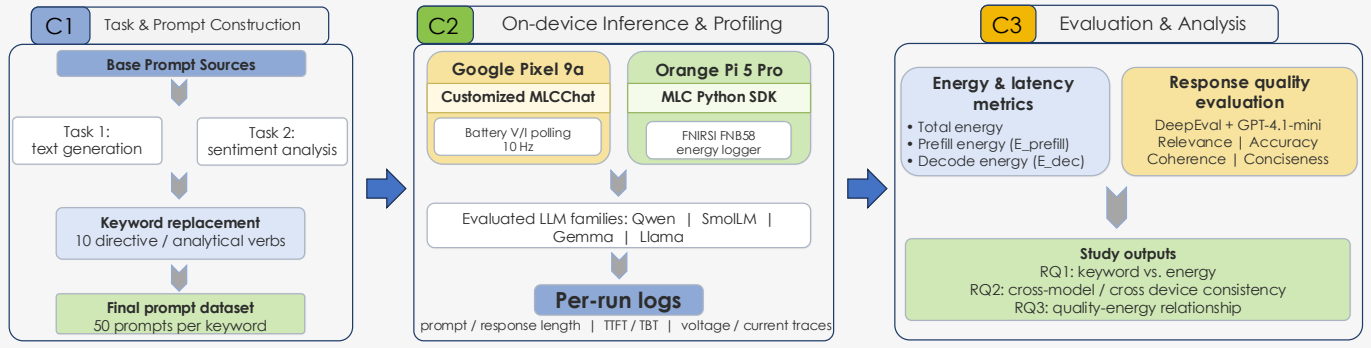

Fig. 1: Overview of the experimental pipeline. Prompts are constructed from text generation and sentiment analysis tasks through keyword replacement, then executed on two edge platforms, Google Pixel 9a | Android 15 and Orange Pi 5 Pro | Ubuntu 22.04, for on-device inference and energy profiling.

B. Prompt Engineering and Energy Awareness

Prompting strategies directly influence reasoning depth and token generation. Chain-of-Thought [16] enhances multi-step reasoning at the cost of longer decoding sequences. Automatic Prompt Engineering (APE) [17] and Active-Prompt [18] search or select effective prompts to guide model behavior. While these studies focus on improving accuracy, efficiency, or controllability, few consider their impact on energy usage. Our work extends this line by empirically measuring how imperative keyword choices in prompts affect *prefill* and *decode* energy consumption on a smartphone, introducing a sustainability perspective to prompt engineering and bridging the gap between linguistic design and device-level energy efficiency.

III. Background and Motivation

A. Background

Large Language Models (LLMs) such as ChatGPT, Gemini, and Claude rely on massive cloud-based GPU and TPU clusters for real-time inference. While this setup enables scalability and continuous updates, it raises concerns about **privacy**, **latency**, and **energy consumption**. Global data centers hosting AI workloads are projected to consume nearly 1,000 TWh of electricity annually by 2030.

To mitigate these challenges, recent progress in quantization, pruning, and efficient transformer architectures has enabled *on-device LLMs*, compact models such as `Phi-3-mini`, `Qwen-1.5`, and `LLaMA 3.2 1B/3B`—that can run locally on smartphones or embedded NPUs. These on-device deployments improve privacy and responsiveness but introduce a new bottleneck: **energy efficiency**. Autoregressive inference involves two main phases: *prefill* and *decode*, where the latter dominates both runtime and power usage. Empirical profiling shows that total energy scales almost linearly with output length and decoding latency, making energy consumption a critical constraint for edge deployment.

B. Motivation

Prior efforts to improve energy efficiency have focused on hardware- or model-level techniques, such as quantization, pruning, or kernel optimization, which require retraining and platform-specific tuning. However, a lightweight and overlooked alternative lies in **prompt design**. Linguistic choices such as "*generate*," "*explain*," or "*list*" can alter reasoning depth, token length, and activation patterns—thereby affecting energy use even under the same model and hardware configuration.

Despite growing evidence that prompt phrasing influences computation, existing studies are largely theoretical or simulation-based, lacking validation on real devices. This work fills that gap by providing the first empirical analysis linking *imperative keyword choices* to measured power consumption during on-device LLM inference. Our findings highlight prompt engineering as a simple yet effective lever for achieving *energy-efficient and privacy-preserving* intelligence at the edge.

IV. Experimental Design

A. Research Objective and Questions

Building on Section III, our objective is to quantify how linguistic variations, particularly the choice of imperative *keywords*, affect energy consumption of on-device LLMs. This study is guided by three research questions:

- **RQ1:** How do different prompt keywords affect overall energy consumption of on-device LLM inference?
- **RQ2:** Are observed keyword-driven energy patterns consistent across different LLM models and devices?
- **RQ3:** How is response quality affected by prompt keywords, and how does it correlate with energy consumption?

B. Hardware Setup, Model and Experimental Pipeline

Hardware Setup: Experiments were conducted on an Orange Pi 5 Pro running Ubuntu 22.04 and a rooted Google Pixel 9a. These two devices each have different capabilities that became apparent in the study. Orange Pi 5 Pro excelled in prefill, completing its prefill phase in an average of 115ms, compared to Pixel 9a, which completed its prefill phase in an average of 87,000ms (~750x that of Orange Pi 5 Pro), a concerning value that suggests the GPU was not being used. However, decode rate was similar across devices, 114ms TBT for Orange Pi 5 Pro, and 100ms TBT for Pixel 9a. To reduce interference from

unrelated system activity, all measurements on Orange Pi 5 Pro were performed in multi-user mode. On Pixel 9a, CPU frequencies were fixed at 1328 kHz for Cores 0–3, 1418 kHz for Cores 4–6, and 2294 kHz for Core 7. To maintain stable power measurements, thermal throttling, adaptive brightness, and background services were disabled, and screen brightness was fixed at the minimum level to reduce display-related noise. The script used is detailed in Appendix A.

Models and Experimental Pipeline: The end-to-end measurement workflow is illustrated in Figure 1. We evaluated models from Qwen, SmolLM, Gemma, and Llama families to ensure that our observations are consistent across architectures. Most selected models were matched to have similar parameter sizes and quantization settings, while one smaller model was additionally included to study the effect of model size. All models except Gemma-3-1B are instruct models. Gemma-3-1B was selected to test how a different type of model would be affected by different keywords.

C. Tasks, Prompts, and Data Collection

To relate linguistic structure to energy consumption, we evaluate two representative NLP tasks capturing open-ended and analytical behavior:

Text Generation: Prompts adapted from the Alpaca-GPT4 dataset [19] (creative writing, explanation). 50 prompts were chosen, and the keyword was replaced with 10 selected directive verbs. Previously, we chose 50 different prompts for each keyword instead of using the same 50 prompts for each keyword. However, due to inherent biases in the usage of each keyword in the Alpaca-GPT4 dataset, this led to results that were suspiciously consistent across environments. The text generation prompts used in this paper only differ from each other by keyword, isolating its impact.

Sentiment Analysis: Binary sentiment analysis prompts synthesized using Yelp review data [20] with analytical verbs (*determine*, *label*, etc.), representing short, deterministic reasoning. These prompts use the same 50 yelp reviews, combined with a short sentence that prompts binary sentiment analysis. Similarly to text generation, this was also done to isolate the impact of keywords.

Some sample prompts that we used are available in Appendix B

Prompts: For each keyword, we selected 50 prompts, resulting in 1,000 prompts in total (500 per task) and 5,000 prompt-model runs across 5 models. Additionally, each run was averaged across 3 trials to ensure experimental consistency and mitigate randomness. To verify that keyword replacement does not substantially change prompt difficulty, we evaluated both the original and rewritten Text Generation prompts using the NVIDIA prompt-task-and-complexity classifier [21]. As shown in Figure 2, the two score distributions largely overlap, suggesting that the rewritten prompts preserve the overall complexity profile of the original ones. Quantitatively, the matched original prompts have a mean complexity score of 0.3048, compared to 0.3199 for the rewritten prompts, corresponding to a small average shift of +0.0151 on a 0–1 scale. Furthermore, after averaging across rewritten variants, the prompt-level scores remain highly correlated with the original prompts (Pearson $r = 0.942$, Spearman $\rho = 0.944$). Together, these results show that keyword replacement introduces only a minor change in classifier-assigned complexity while preserving the relative difficulty structure of the prompt set.

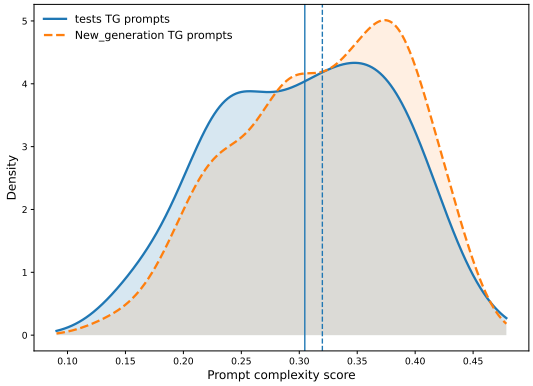

Fig. 2: Prompt-complexity distribution comparison between the original and keyword-replaced text-generation prompt dataset.

Data Collection: For Orange Pi 5 Pro, we used the Python API to run the models and FNIRSI FNB58 to collect energy data. The output was parsed using a third-party data logger [22]. For Pixel 9a, we ran a customized `MLCChat` that integrates logging into the inference pipeline by measuring power draw from the battery while the phone is discharging. Responses are capped at 1,000 tokens to stop infinite loops occasionally observed in small quantized models. Responses that exceed the token limit are excluded from our final data.

Per-run metrics include: *prompt length* (L_p), *response length* (L_r), *TTFT*, *TBT*, and instantaneous *voltage/current* traces sampled at 10 Hz. Total energy is computed as:

$$E = \sum_{i=1}^{N} I_i V_i \Delta t, \qquad \Delta t = 0.1 \text{ s},$$

Where I_i and V_i are instantaneous current and voltage respectively, and Δt is the sampling delay. and logged separately for *prefill* (E_{prefill}) and *decoding* (E_{dec}). To prevent memory accumulation, the model instance is reloaded before each run.

We found that for each model, energy consumed during the decode phase is directly proportional to response length. This reveals that each token consumes roughly the same amount of energy, suggesting that MLC operates on a dense inference pipeline rather than a sparsity aware pipeline, where it would be expected for certain tokens to require heavier calculations. Since each token costs roughly the same amount of energy, differences observed must be a result of response length, showing that changing a single word in prompting can cause model to end its response early, and reducing energy consumption.

D. Accuracy Evaluation

To evaluate response quality, we used the DeepEval Python library [23] with GPT-4.1-mini as an LLM judge to perform

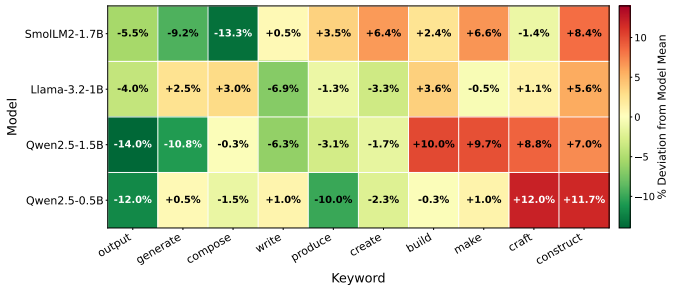

Fig. 3: **Text generation** decode energy deviation from model mean (%) across classification-related keywords on the Orange Pi 5 Pro. Values represent per-keyword deviation from each model's mean decode energy.

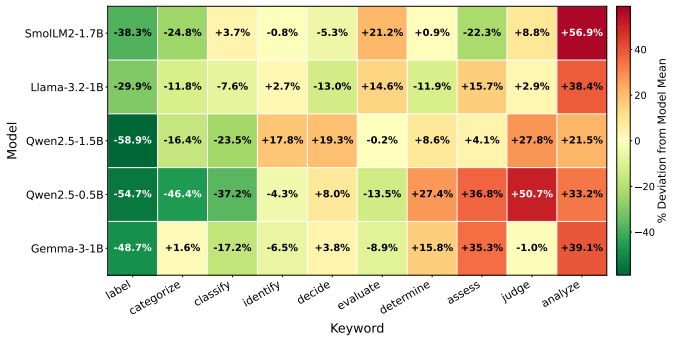

Fig. 4: **Sentiment analysis** decode energy deviation from model mean (%) across classification-related keywords on the Orange Pi 5 Pro. Values represent per-keyword deviation from each model's mean decode energy.

G-Eval on each input–output pair [24]. Model responses were assessed according to the following criteria:

1) **Relevance:** whether the output meaningfully addresses the questions, tasks, or instructions expressed in the input prompt.
2) **Accuracy:** whether the factual claims in the output are correct, verifiable, and consistent with established knowledge or information explicitly provided in the input.
3) **Coherence:** whether the output is logically organized, internally consistent, and easy to follow as a unified response.
4) **Conciseness:** whether the output conveys the required information efficiently, without unnecessary detail beyond what is needed to satisfy the intent of the input.

The full DeepEval setup is available in Appendix C

Although the judge model may implicitly consider factual accuracy across multiple criteria, we found that this overlap has minimal effect on the significance of the overall score.

V. Empirical Study and Results

Building upon the experimental framework described in Section IV, this section presents empirical findings addressing our three research questions (RQ1–RQ3). We analyze how linguistic variations, particularly keywords, influence the energy consumption of on-device LLMs across 5 models and two task types (text generation and sentiment analysis). Unless otherwise specified, all energy values refer to the decoding phase (E_{dec}).

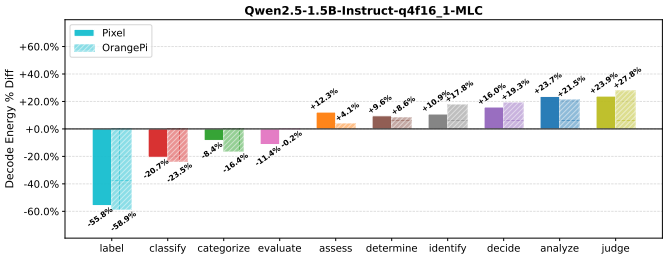

Fig. 5: **Sentiment analysis** decode energy deviation from model mean for Pixel 9a and Orange Pi 5 Pro, shown for Qwen2.5-1.5B

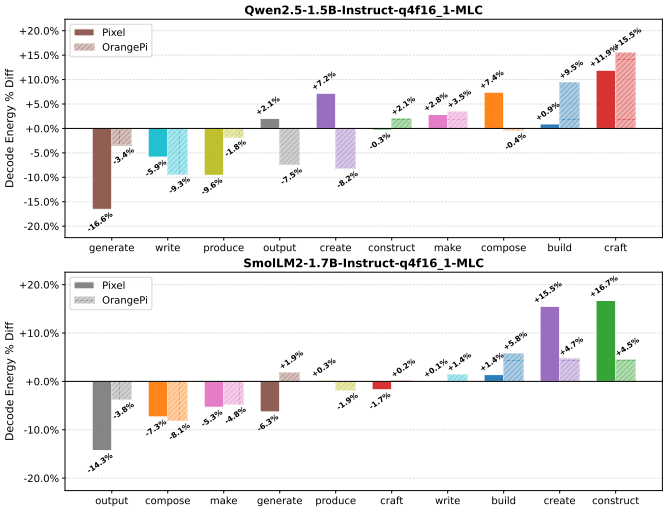

Fig. 6: **Text generation** decode energy deviation from model mean for Pixel 9a and Orange Pi 5 Pro, shown for Qwen2.5-1.5B and SmolLM2-1.7B.

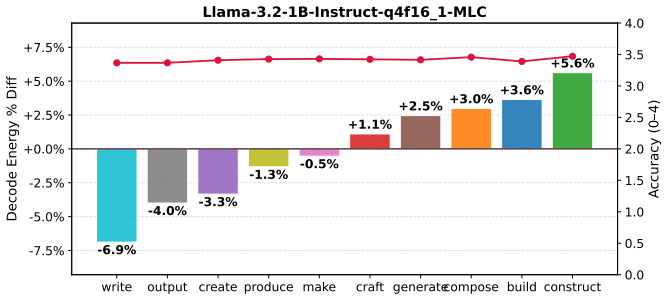

Fig. 7: **Text generation** decode energy % deviation from model mean and output accuracy on the Orange Pi 5 Pro, shown for Llama-3.2-1B

All figures show how the mean energy consumption for each keyword compares to the mean energy consumption across the whole model, represented as a percentage deviation.

Text generation data collected from the Gemma-3-1B model

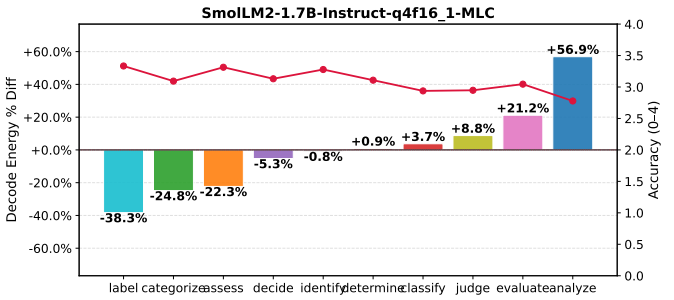

Fig. 8: **Sentiment analysis** decode energy % deviation from model mean and output accuracy on the Orange Pi 5 Pro, shown for SmolLM2-1.7B

was excluded from graphs due to nearly all responses exceeding the token limit, due to the infinite repetition of tokens.

A. Consistency across repeat runs

To ensure experimental consistency, we ran each prompt 3 times and found the average value. We found that the average standard deviation of energy consumption across each prompt was 55J, while the average energy cost during decode was 219J. When considering just text generation prompts, the average standard deviation rose to 79J, while the average energy cost rose to 371J. When considering only sentiment analysis prompts, the average decode energy cost decreased to just 67J, meaning despite the lower average standard deviation of 31J, variability is extremely high. This shows high variability between runs, expected from the randomness of LLM behavior.

A different metric is needed to effectively measure consistency across runs. By taking the percentage deviation from each keyword from its model mean, we can compare the results of each model and repeat using the Pearson correlation coefficient. Taking the average correlation across repeats and models, we find that for text generation, the correlation of energy consumption across runs averages to 0.28, which is still significantly variable, but shows slight consistency. For sentiment analysis, on the other hand, the correlation averages to 0.61, representing moderate correlation.

These results show that despite the inherent variability of LLMs, the results of keywords across runs hold much better than would be expected of completely random results.

B. RQ1: How do different prompt keywords affect the overall energy consumption of on-device LLM inference?

Text Generation: Figure 3 reveals that using an energy efficient keyword can save up to 14% energy during the decode phase of text generation tasks ("output" keyword in Qwen2.5-1.5B). Meanwhile, an inefficient choice of keyword can increase consumption by up to 11.7% ("construct" keyword in Qwen2.5-0.5B).

We define keyword sensitivity as the standard deviation of the percentage deviations per keyword in a model, and represents how sensitive a language model is to changes in keywords. In the Qwen models, the keyword sensitivity was 8.28% for Qwen2.5-1.5B and 7.31% for Qwen2.5-0.5B. However, for Llama-3.2-1B, the keyword sensitivity was only 3.70%, significantly less than the Qwen models. Of the models in this experiment, the order from most to least sensitive is: Qwen2.5-1.5B, 0.5B (8.28% and 7.31%), SmolLM2-1.7B (6.83%), Llama-3.2-1B (3.70%).

Sentiment Analysis: Figure 4 shows that there is more promise to reduce energy consumption in sentiment analysis, since models display higher keyword sensitivity. Across all 5 models, the keyword "label" performed very well consistently, saving between 29.9% and 58.9% decode energy from the mean in all models, suggesting that it could be heavily optimized in other language models as well.

Energy savings in sentiment analysis are much greater than in text generation when viewing percentages. This is likely because sentiment analysis generates shorter responses, so percentage changes are amplified by shorter responses. Aside from the "label" keyword, the "categorize" keyword also performed better than average in all 5 models, but it especially shined in Qwen2.5-0.5B.

Keyword sensitivity is much higher in sentiment analysis, though this is because percentage changes are higher. Additionally, sensitivity rankings stayed relatively consistent compared with text generation, both having Llama-3.2-1B with the lowest sensitivity of 18.31%, and a Qwen model (Qwen2.5-0.5B) with the highest sensitivity of 35.46%.

Insight 1: *Models differ in energy sensitivity, with certain model families, like Qwen, being much more sensitive to keywords, resulting in larger energy savings.*

The full rankings are: Qwen2.5-0.5B (35.46%), SmolLM2-1.7B (25.27%), Qwen2.5-1.5B (25.11%), Gemma-3-1B (24.11%), and Llama-3.2-1B (18.31%). As opposed to text generation, the sensitivities of each model, discounting the highest and lowest, hover within a single percentage point from 25%, compared to a wide range of sensitivities.

C. RQ2: Are the observed keyword-driven energy patterns consistent across different LLM models and devices?

TABLE I: Pairwise Pearson Correlation Matrix-Text Generation

Model	**(1)**	**(2)**	**(3)**	**(4)**
(1) SmolLM2-1.7B				
(2) Llama-3.2-1B	-0.076			
(3) Qwen2.5-1.5B	0.511	0.528		
(4) Qwen2.5-0.5B	0.238	0.503	0.637	

Consistency Across Models: Using the pairwise Pearson coefficient of the ranks of keywords across models, we get an average coefficient of 0.390 in text generation, and an average $\rho = 0.658$ in sentiment analysis, showing that there is moderate correlation across models. Additionally, some patterns stand out.

Most noticeably, the two Qwen models had a Pearson correlation of $\rho = 0.637$ in text generation, and $\rho = 0.836$ in sentiment analysis, significantly higher than correlations

TABLE II: Pairwise Pearson Correlation Matrix-Sentiment Analysis

Model	(1)	(2)	(3)	(4)	(5)
(1) SmolLM2-1.7B					
(2) Llama-3.2-1B	0.786				
(3) Qwen2.5-1.5B	0.604	0.607			
(4) Qwen2.5-0.5B	0.482	0.589	0.836		
(5) gemma-3-1b	0.492	0.733	0.709	0.746	

between other models, showing that trends are more consistent in the same model family, due to having the same training data set and tokenizer. However, in text generation, Llama-3.2-1B and SmolLM2-1.7B display a negative, near-zero correlation of -0.076, suggesting the possibility of outlier models which behave differently when prompted with different keywords than other models. Aside from this outlier, the average Pearson coefficients reflect moderate correlation between models, suggesting that for the models tested, keyword performance trends are generally persistent, especially in sentiment analysis.

Insight 2: *Keywords that perform well on one model generally tend to perform above average in other models, indicating the correlation between keyword and energy consumption is relatively consistent.*

Consistency Across Devices: Figure 5 shows that sentiment analysis results have near perfect correlation across devices, with Qwen2.5-1.5B having a Pearson $\rho = 0.973$. This proves that for sentiment analysis prompts, the same keywords exhibit the same trends across devices.

The same comparison was done for text generation. While SmolLM2-1.7B had $\rho = 0.707$, suggesting moderate to high correlation, Qwen2.5-1.5B had a Pearson $\rho = 0.411$, only moderate correlation. The Spearman coefficients were also significantly lower. Additionally, Figure 6 shows only slight correlation between the devices, significantly smaller than in sentiment analysis. This suggests that in text generation, inherent randomness in the open ended prompts can lead to deviations across devices, as well as across models as seen above.

Insight 3: *Due to the open ended nature of text generation prompts, their energy consumption can fluctuate across environments, whether it be model architectures or devices. By contrast, sentiment analysis consumption stays relatively consistent.*

D. RQ3: How is response quality affected by prompt keywords, and how does it correlate with energy consumption?

Text Generation: Accuracy is measured on a scale from 0–4, with 2 being a passing score (criteria described above and in Appendix C). In text generation, the Spearman correlation between the rankings by energy consumption, and the rankings by accuracy average to 0.48 for text generation, showing moderate positive correlation. Spearman correlation is used because accuracy is on a different scale, and is not expected to be linear with energy consumption.

The positive correlation suggests that replacing the keyword to favor energy efficiency does negatively impact the accuracy of the response given. However, upon reviewing Figure 7, it shows that response accuracy decreases very little when using a more efficient keyword, decreasing by only 0.1 points of accuracy when using the most energy consuming and most efficient keywords. Other models reveal similar trends, with accuracy remaining largely unaffected across keywords. This suggests that despite the tradeoff between accuracy and energy cost, in reality, the cost in accuracy is negligible when replacing only the keyword.

Sentiment Analysis: In sentiment analysis, however, the correlations averaged to -0.68, which is a moderate inverse correlation. This is also apparent in Figure 8, where accuracy trends downward as energy consumption increases. This inverse correlation suggests longer responses can be prone to errors, though it could also be an effect of conciseness being a metric in the accuracy evaluation. Human analysis of the responses is needed to confirm this.

VI. Future Work

Our current study focuses on an empirical analysis of how prompt keywords affect the energy consumption of on-device LLM inference. While our findings provide initial evidence of a measurable correlation between linguistic structure and power usage, several open questions and opportunities remain for future exploration.

A. Cross-Device and Cross-Model Generalization

Our present experiments focus on two mobile platforms and four compact LLM variants. Future work will extend this to a broader set of devices, including smartphones, embedded NPUs, and edge accelerators such as Jetson or Coral TPUs. Additionally, the prompt dataset will be expanded to cover a more diverse range of keywords, which may reveal patterns when conducting linguistic analysis. Comparing different architectures and hardware configurations will reveal how linguistic-computational correlations generalize across hardware generations and instruction sets.

VII. Conclusion

Prompt formulation has a noticeable impact on the energy footprint of on-device LLM inference. Our empirical results suggest that prompt engineering is a practical, user-level strategy for improving energy efficiency in LLMs. Persistent trends in the energy efficiency of keywords suggest that they have a direct impact on the performance of LLMs, showing the potential of prompt construction on saving energy. Additionally, it suggests other aspects of prompting may also serve as easy ways to reduce resource usage of LLMs. As AI becomes more prevalent, reducing its energy consumption becomes increasingly important to ensure sustainability, and prompt engineering is an easily-implementable way of achieving that goal.

REFERENCES

[1] Meta AI, "The llama 3 herd of models," https://ai.meta.com/research/publications/the-llama-3-herd-of-models/, 2024, accessed 2025-11-01.

[2] Microsoft Research, "Phi-3 technical report: A highly capable language model," 2024.

[3] MLC AI, "Mlc-llm: Universal deployment of llms," https://github.com/mlc-ai/mlc-llm, 2023, accessed 2025-11-01.

[4] NVIDIA, "Tensorrt-llm," https://github.com/NVIDIA/TensorRT-LLM, 2023, accessed 2025-11-01.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[6] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[7] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[8] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," in *ACL*, 2019.

[9] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," in *NeurIPS*, 2022.

[10] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," *arXiv preprint arXiv:2309.06180*, 2023.

[11] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *ICML*, 2023.

[12] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *NeurIPS*, 2023.

[13] C. Liu *et al.*, "Mobilellm: Optimizing sub-billion parameter language models for on-device use," *arXiv preprint arXiv:2402.14905*, 2024.

[14] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," *arXiv preprint arXiv:2007.03051*, 2020.

[15] A. S. Luccioni, T. Viguier, A. Hernández-García *et al.*, "Estimating the carbon footprint of BLOOM, a 176b parameter language model," *Journal of Machine Learning Research*, vol. 24, no. 169, 2023.

[16] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022.

[17] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," *arXiv preprint arXiv:2211.01910*, 2022.

[18] S. Diao, P. Wang, Y. Lin, R. Pan, X. Liu, and T. Zhang, "Active prompting with chain-of-thought for large language models," *arXiv preprint arXiv:2302.12246*, 2023.

[19] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. Hashimoto, "Stanford alpaca: Code and data," https://github.com/tatsu-lab/stanford_alpaca, 2023, accessed 2025-11-01.

[20] Yelp, Inc., "Yelp open dataset," https://business.yelp.com/data/resources/open-dataset/, 2024, accessed 2025-11-01.

[21] NVIDIA, "Nemocurator prompt task and complexity classifier," https://huggingface.co/nvidia/prompt-task-and-complexity-classifier, 2024, hugging Face model card, version 1.1, accessed March 19, 2026.

[22] Baryluk, "Fnirsi usb power data logger," https://github.com/baryluk/fnirsi-usb-power-data-logger, 2026, gitHub repository, accessed March 18, 2026.

[23] "Deepeval python library," GitHub repository, 2026, available at: https://github.com/confident-ai/deepeval, accessed March 18, 2026.

[24] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-eval: NLG evaluation using gpt-4 with better human alignment," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2511–2522. [Online]. Available: https://aclanthology.org/2023.emnlp-main.153/

APPENDIX A
PIXEL 9A PROFILING LOCKDOWN SCRIPT

```
#!/system/bin/sh
# profiling_lockdown.sh
# Pixel 9a profiling lockdown mode (with 80% freq cap)

echo "[*] Entering PROFILING LOCKDOWN MODE"

echo "[CPU] Locking CPU frequencies to ~80% of hardware max"
for cpu in /sys/devices/system/cpu/cpu[0-3]*; do
    cpuinfo_max="$cpu/cpufreq/cpuinfo_max_freq"
    min_file="$cpu/cpufreq/scaling_min_freq"
    max_file="$cpu/cpufreq/scaling_max_freq"
    gov_file="$cpu/cpufreq/scaling_governor"

    if [ -f "$cpuinfo_max" ] && [ -f "$min_file" ] && [ -f "$max_file" ] && [ -f "$gov_file" ]; then
        hw_max=$(cat $cpuinfo_max)
#       target=$((hw_max * 80 / 100))
        target=1328000

        echo userspace > "$gov_file" 2>/dev/null
        echo $target > "$min_file" 2>/dev/null
        echo $target > "$max_file" 2>/dev/null

        echo "$(basename $cpu) locked to $target Hz (~80% of hardware max $hw_max Hz)"
    fi
done
for cpu in /sys/devices/system/cpu/cpu[4-6]*; do
    cpuinfo_max="$cpu/cpufreq/cpuinfo_max_freq"
    min_file="$cpu/cpufreq/scaling_min_freq"
    max_file="$cpu/cpufreq/scaling_max_freq"
    gov_file="$cpu/cpufreq/scaling_governor"

    if [ -f "$cpuinfo_max" ] && [ -f "$min_file" ] && [ -f "$max_file" ] && [ -f "$gov_file" ]; then
        hw_max=$(cat $cpuinfo_max)
#       target=$((hw_max * 80 / 100))
        target=1418000

        echo userspace > "$gov_file" 2>/dev/null
        echo $target > "$min_file" 2>/dev/null
        echo $target > "$max_file" 2>/dev/null

        echo "$(basename $cpu) locked to $target Hz (~80% of hardware max $hw_max Hz)"
    fi
done
for cpu in /sys/devices/system/cpu/cpu[7-7]*; do
    cpuinfo_max="$cpu/cpufreq/cpuinfo_max_freq"
    min_file="$cpu/cpufreq/scaling_min_freq"
    max_file="$cpu/cpufreq/scaling_max_freq"
    gov_file="$cpu/cpufreq/scaling_governor"

    if [ -f "$cpuinfo_max" ] && [ -f "$min_file" ] && [ -f "$max_file" ] && [ -f "$gov_file" ]; then
        hw_max=$(cat $cpuinfo_max)
#       target=$((hw_max * 80 / 100))
        target=2294000

        echo userspace > "$gov_file" 2>/dev/null
        echo $target > "$min_file" 2>/dev/null
        echo $target > "$max_file" 2>/dev/null

        echo "$(basename $cpu) locked to $target Hz (~80% of hardware max $hw_max Hz)"
    fi
done

mali_path="/sys/class/devfreq/20c00000.callisto"
target=560000000
```

```
echo "[GPU] Locking GPU frequency to $target Hz"
echo userspace > "$mali_path/governor" 2>/dev/null
echo target > "$mali_path/min_freq" 2>/dev/null
echo target > "$mali_path/max_freq" 2>/dev/null
echo "GPU locked to $target Hz"

# --- Disable Doze / Deep idle ---
dumpsys deviceidle disable
echo "[+] Doze / deep idle disabled"

# --- Thermal throttling: disable all zones ---
for zone in /sys/class/thermal/thermal_zone*; do
    mode="$zone/mode"
    if [ -f "$mode" ]; then
        echo disabled > "$mode" 2>/dev/null
    fi
done
echo "[+] Thermal throttling disabled"

# --- System features ---
pm disable-user --user 0 com.google.android.gms >/dev/null 2>&1
echo "[+] Google Play Services fully disabled"

cmd notification set_dnd none
echo "[+] Push notifications disabled"

settings put system screen_brightness_mode 0
settings put system screen_brightness 0
echo "[+] Adaptive brightness off, screen set to minimum"

settings put secure location_mode 0
echo "[+] Location services disabled"

echo "[*] Lockdown complete. Ready for profiling."
```

APPENDIX B
SAMPLE LLM PROMPTS

```
   Text Generation:

Produce 10 multiple choice questions about the human circulatory system
Produce 2 strategies for reducing stress.
Produce 4 essential questions on the topic of legal regulation of AI.
Produce 5 possible slogans for an online shoe company
Produce a 4 step plan for tackling a problem. Homelessness

Create 10 multiple choice questions about the human circulatory system
Create 2 strategies for reducing stress.
Create 4 essential questions on the topic of legal regulation of AI.
Create 5 possible slogans for an online shoe company
Create a 4 step plan for tackling a problem. Homelessness



   Sentiment Analysis:

Classify the sentiment of the following Yelp review as positive or negative: Wow! Yummy, different, delicious. Our favorite is the lamb curry and korma. With 10 different kinds of naan!!! Don't let the outside deter you (because we almost changed our minds)...go in and try something new! You'll be glad you did!
Classify the sentiment of the following Yelp review as positive or negative: Amazingly amazing wings and homemade bleu cheese. Had the ribeye: tender, perfectly prepared, delicious. Nice selection of craft beers. Would DEFINITELY recommend checking out this hidden gem.
Classify the sentiment of the following Yelp review as positive or negative: Locals recommended Milktooth, and it's an amazing jewel of Indianapolis. I'm glade I had the chance to experience this.
Classify the sentiment of the following Yelp review as positive or negative: Love going here for happy hour or dinner! Great patio with fans to beat the StL heat! Also...very accomodating at this location. I like the Veal Milanese but with mixed greens instead of pasta! they'll modify the menu to suit your taste!
Classify the sentiment of the following Yelp review as positive or negative: Great place for breakfast! I had the waffle, which was fluffy and perfect, and home fries which were nice and smashed and crunchy. Friendly waitstaff. Will definitely be back!

Decide whether the sentiment expressed in this Yelp review is positive or negative: Wow! Yummy, different, delicious. Our favorite is the lamb curry and korma. With 10 different kinds of naan!!! Don't let the outside deter you (because we almost changed our minds)...go in and try something new! You'll be glad you did!
Decide whether the sentiment expressed in this Yelp review is positive or negative: Amazingly amazing wings and homemade bleu cheese. Had the ribeye: tender, perfectly prepared, delicious. Nice selection of craft beers. Would DEFINITELY recommend checking out this hidden gem.
Decide whether the sentiment expressed in this Yelp review is positive or negative: Locals recommended Milktooth, and it's an amazing jewel of Indianapolis. I'm glade I had the chance to experience this.
Decide whether the sentiment expressed in this Yelp review is positive or negative: Love going here for happy hour or dinner! Great patio with fans to beat the StL heat! Also...very accomodating at this location. I like the Veal Milanese but with mixed greens instead of pasta! they'll modify the menu to suit your taste!
Decide whether the sentiment expressed in this Yelp review is positive or negative: Great place for breakfast! I had the waffle, which was fluffy and perfect, and home fries which were nice and smashed and crunchy. Friendly waitstaff. Will definitely be back!



   NOTE: The following prompts produced biased results in the original experiment

Write a 1 sentence summary of digital marketing.
Make 10 sentences with the word 'big'.
Write 7 words related to the word "party".
Create 4 antonyms for the given word Word: Strident
Construct a 3-note melody.
```

APPENDIX C
GEVAL METRIC DEFINITIONS

```
GEval(
    name="Relevance",
    criteria="Evaluate whether the actual output meaningfully attempts to address the questions, tasks, or instructions expressed in the input prompt.",
    evaluation_steps=[
        "Decompose the input prompt into all explicit questions, tasks, or instructions",
        "Identify any clearly implied or dependent sub-requests required to fulfill the prompt",
        "Check whether the actual output attempts to address each identified component",
        "Penalize missing, ignored, or substituted prompt components",
        "Do NOT evaluate factual correctness, depth, or writing quality-only whether an attempt is made to respond to the prompt"
    ],
    model="gpt-4.1-mini",
    evaluation_params=[LLMTestCaseParams.INPUT, LLMTestCaseParams.ACTUAL_OUTPUT]
),
GEval(
    name="Accuracy",
    criteria="Evaluate whether the factual claims in the actual output are correct, verifiable, and consistent with established knowledge or information explicitly provided in the input.",
    evaluation_steps=[
        "Identify all factual claims made in the actual output",
        "Verify each claim against reliable general knowledge or facts stated in the input",
        "Penalize claims that are clearly false or contradict known facts or the input",
        "Penalize incorrect or incomplete answers to explicit factual questions in the input",
        "Do NOT penalize appropriately qualified uncertainty (e.g., 'may', 'depends') when facts are ambiguous or unspecified",
        "Do NOT evaluate writing style, coherence, verbosity, or relevance-only factual correctness"
    ],
    model="gpt-4.1-mini",
    evaluation_params=[LLMTestCaseParams.INPUT, LLMTestCaseParams.ACTUAL_OUTPUT]
),
GEval(
    name="Coherence",
    criteria="Evaluate whether the actual output is logically structured, internally consistent, and easy to follow as a unified response.",
    evaluation_steps=[
        "Identify the main claim, explanation, or narrative the actual output is presenting",
        "Check whether ideas progress in a clear and logical order (e.g., cause-effect, step-by-step, or narrative flow)",
        "Detect internal contradictions or statements that undermine earlier claims and penalize them heavily",
        "Ensure transitions between ideas are clear and do not cause confusion or logical jumps",
        "Penalize tangents or digressions only if they disrupt logical flow or comprehension",
        "Do NOT evaluate factual correctness relative to the input or external knowledge-only internal clarity and consistency"
    ],
    model="gpt-4.1-mini",
    evaluation_params=[LLMTestCaseParams.INPUT, LLMTestCaseParams.ACTUAL_OUTPUT]
),
GEval(
    name="Conciseness",
    criteria="Evaluate whether the actual output communicates the required information efficiently, using no more words than necessary given the intent of the input.",
    evaluation_steps=[
        "Identify the core information or task required by the input prompt",
        "Check whether the actual output includes only information necessary to fulfill that task",
        "Flag redundancy, repetition, filler phrases, or tangential details that do not add meaning",
        "Do NOT penalize length if the input explicitly requests detailed explanation, reasoning, or creativity",
        "Do NOT penalize content that materially improves clarity, correctness, or completeness",
        "Prefer answers that match the shortest clear version an expert would reasonably give"
    ],
    model="gpt-4.1-mini",
    evaluation_params=[LLMTestCaseParams.INPUT, LLMTestCaseParams.ACTUAL_OUTPUT]
)
```