# Energy-Aware Benchmarking of Generative Information Extraction

**Anonymous ACL submission**

## Abstract

Using Large Language Models (LLMs) for Information Extraction (IE) inherently involves transforming diverse textual inputs into strictly formatted outputs. Guaranteeing this structural validity introduces a hidden energy tax driven by constrained decoding mechanisms, serialisation formats, and underlying serving configurations. Consequently, standard token-level metrics fail to capture the true energy cost of generative IE, which also cannot be predicted by model size alone. To address this, we introduce the first dedicated benchmark evaluating the energy-accuracy trade-offs of generative IE. Our framework establishes task-aligned energy metrics, including Energy per True Positive ($J/TP$) and the F1-Energy Ratio ($F_1/\text{kJ}$). Through comprehensive ablation across architectures, decoding regimes, and serving-layer limits, we demonstrate that efficiency is governed by complex parameter interactions. We translate these findings into actionable configuration guidelines, equipping practitioners to maximise predictive accuracy, optimise energy yield, or achieve a sustainable Pareto balance. The source code and evaluation results are available on GitHub[1].

## 1 Introduction

Information Extraction (IE) is rapidly shifting from highly efficient, fine-tuned encoders to zero-shot Large Language Models (LLMs) (Ni et al., 2025). While LLMs offer immense flexibility, their widespread deployment raises serious environmental concerns (Schwartz et al., 2020). Crucially, generative IE represents a special case of LLM inference: it relies on mapping long, heterogeneous inputs to very short, structurally rigid, and parseable outputs (Zhang et al., 2025). This structural disparity exposes a critical lack of energy-aware benchmarks tailored to the operational realities of constrained generation.

Equating smaller model architectures with proportional energy savings is an oversimplification, as the efficiency of generative IE operates as a systems-level property. Since structured extraction tasks (NER, RE, EE) demand strict schema adherence (Politov et al., 2025), assessing their actual energy overhead requires explicitly integrating output formats, constrained decoding mechanisms, and serving-layer configurations into the evaluation framework. Enforcing these constraints imposes a "computational tax" that alters energy consumption independently of raw token volume. Consequently, standard token-level energy metrics, which treat all generated text equally, are fundamentally insufficient for capturing the true GPU inference cost of structured extraction.

To map these unique inefficiencies, we establish the first dedicated benchmark for sustainable IE. Specifically, we make three core contributions. First, we introduce a comprehensive energy-aware benchmark for NER, RE, and EE spanning multiple model scales, output formats, decoding regimes, serving configurations, and GPU generations. Second, we define a standardised methodology using task-aligned energy metrics, such as the F1-Energy Ratio ($F_1/\text{kJ}$) and Energy per True Positive ($J/TP$), thereby demonstrating why conventional token-level metrics fail for structured extraction. Finally, we provide an empirical deployment analysis that translates these interactions into actionable guidelines (See Table 1). Specifically, our evaluation investigates whether model size is a reliable predictor of energy efficiency; whether supervised encoders remain preferable to zero-shot decoders; how output formatting and constrained decoding affect the energy-accuracy trade-off; and the extent to which serving-layer parameters alter the efficiency frontier.

[1] https://anonymous.4open.science/r/Energy-Benchmarking-of-IE-FB11

## 2 Related Work

**Encoder only models** Historically, IE tasks relied on fully supervised, encoder-only architectures, with models like mBERT (Pires et al., 2019) and XLM-R (Conneau et al., 2020) establishing strong baselines. Complex structured prediction tasks required specialised architectures, such as DREEAM for document-level RE (Ma et al., 2023) and argument-linking networks for EE (Ebner et al., 2020). In low-resource scenarios, translation-plus-projection pipelines (García-Ferrero et al., 2022; Poncelas et al., 2023) achieve predictive accuracy but introduce immense computational overhead during the translation step (Politov et al., 2025).

**LLMs for IE tasks** Generative LLMs have shifted the IE paradigm toward zero- and few-shot prompting (Zhang et al., 2025), bypassing translation pipelines with pre-trained multilingual representations (Mo et al., 2025). However, enforcing strict schema adherence (e.g., JSON, DST) (Bourhis et al., 2020) (Shen et al., 2025) necessitates constrained decoding mechanisms (Ruan et al., 2025). Libraries like Outlines (Willard and Louf, 2023) and XGrammar (Dong et al., 2024) guarantee valid outputs but impose a hidden autoregressive computational overhead, making their deployment significantly more resource-intensive than that of single-pass encoders.

**Benchmarks** The NLP community lacks a unified benchmark targeting the intersection of IE and thermodynamic efficiency. Standard IE benchmarks prioritise predictive accuracy (F1) (Yao et al., 2019), (Hu et al., 2020), (Yu et al., 2026), while efficiency benchmarks evaluate raw throughput or memory, ignoring IE's structural complexities (Chitty-Venkata et al., 2025), (Ni et al., 2025). Consequently, no standardised approach simultaneously evaluates both classical encoders and zero-shot decoders to map the Pareto frontier between accuracy and energy.

**Measurement of the energy consumption** Empirical hardware profiling has largely replaced theoretical FLOP counting for measuring deep learning energy consumption (Zhou et al., 2024), (Niu et al., 2025a). Existing energy studies often ignore the complexities of downstream tasks (Jurkschat et al., 2025). Accurate profiling requires robust telemetry; tools like *nvidia-smi* (Yang et al., 2024) and NVML (Huang et al., 2025) have limitations in polling overhead or scalability (Ansari and Ansari, 2025). Data Centre GPU Manager (DCGM) (Wassermann et al., 2026) offers high-fidelity, low-overhead measurements compared to other solutions. Alongside hardware, we standardise the software serving layer on vLLM, utilising its PagedAttention mechanism for high-concurrency throughput, which provides efficiency benefits compared to other popular inference engines, e.g., HuggingFace Text Generation Inference (TGI) (Kolluru, 2025), being the de facto standard. This controls the computational variance introduced by backend optimisations (Pape et al., 2026). Finally, prior LLM energy evaluations rely on generalised Energy-per-token metrics (Wu et al., 2025), (Solovyeva and Castor, 2026), which are insufficient for structured IE tasks and thus necessitate our task-aligned efficiency metrics.

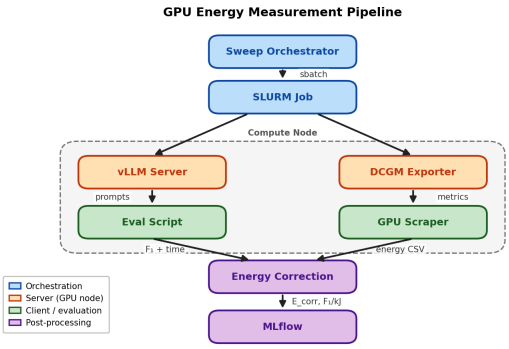


Figure 1: Scheme of the measurement

**Measurement of CO2 and limitations of this approach** Reporting Carbon Dioxide equivalent emissions is problematic for architectural benchmarking (Luccioni et al., 2023). Carbon footprints are tied to the local grid's carbon intensity at the time of computation (Li et al., 2024), causing identical models to yield vastly different footprints based on geographic location (e.g., Germany vs. Norway) (Wu et al., 2022). Because these variables obscure intrinsic algorithmic efficiency, reporting direct energy consumption provides a far more stable, reproducible metric for architectural comparisons.

## 3 Benchmark Design

### 3.1 System Overview

Because the energy efficiency of LLM-based IE is multifaceted, our modular benchmark provides a unified framework for navigating trade-offs between predictive quality (F1) and inference energy across core IE tasks (NER, EE, RE). We evaluate both zero-shot decoders and traditional fine-tuned encoders using shared, task-aligned ef-

ficiency metrics. By systematically isolating variables across tasks, decoding formats, and serving configurations, we map the energy-accuracy frontier. This equips the research community with actionable guidance to optimally configure IE pipelines, whether targeting peak accuracy, maximum efficiency, or a sustainable balance.

### 3.2 Energy Telemetry and Measurement Protocol

Accurate energy profiling requires isolating the *active inference window* from general server overhead. We omit full-system power to focus exclusively on GPU power draw, utilising an NVIDIA DCGM telemetry pipeline that polls the hardware energy counter (`DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION`) at 100 ms intervals. To eliminate initialisation variance, the active window is strictly defined as the continuous period where GPU utilisation exceeds zero. Crucially, we isolate pure computational cost by measuring only direct GPU power draw, explicitly excluding the energy consumption of the host CPU, system RAM, and VRAM. This calibrated estimator successfully excludes model loading and initial cache allocation, but it inherently captures operational realities such as prompt parsing, constrained decoding overhead, and baseline idle power. All configurations were run three times ($N = 3$); the mean coefficient of variation across runs remained below 7%, confirming measurement reproducibility under hardware thermal and scheduling variance.

### 3.3 Task-Aware Energy Metrics

The conventional efficiency baseline, Energy per Generated Token ($E_{\text{Gen}} = \Delta E / N_{\text{gen}}$, for $N_{\text{gen}}$ output tokens), is fundamentally flawed for Information Extraction (IE). It conflates formatting verbosity with utility, rewarding fast hallucinations while penalising structured prompts. Efficiency must instead reflect task-level utility.

We isolate the active inference energy ($\Delta E$, in Joules) by subtracting the cumulative GPU energy recorded at the onset $E(t_{\text{start}}$ of active generation from the total energy recorded at its completion $E(t_{\text{end}}$:

$$\Delta E = E(t_{\text{end}}) - E(t_{\text{start}}) \tag{1}$$

We introduce the **F1-Energy Ratio** ($F_1/\text{kJ}$) ($\hat{F}_1/(\Delta E/1000)$) and its inverse, the Energy-to-F1 Cost ($E_{F1} = \Delta E / \hat{F}_1$), to quantify the thermodynamic yield of accuracy.

Concurrently, **Energy per Extraction** ($E_{\text{extraction}} = \Delta E / \hat{N}_{\text{pred}}$, for $\hat{N}_{\text{pred}}$ predicted spans) measures the raw cost per predicted span. Analysing F1-based efficiency alongside extraction cost reveals whether energy is wasted on over-generation or misclassification.

Finally, to more rigorously approximate the cost of successful extractions, we introduce **Energy per True Positive** ($J/TP$) ($E_{\text{TP}}$). This metric estimates the average energy expended to yield a single correct extraction:

$$E_{\text{TP}} = \frac{\Delta E}{\hat{P} \cdot \hat{N}_{\text{pred}}} \tag{2}$$

where $\hat{P}$ is extraction precision. By severely penalising high-volume, low-precision hedging, $E_{\text{TP}}$ defines the true operational cost of useful outputs.

### 3.4 Taxonomy of Experimental Parameters

To systematically isolate the drivers of inference efficiency and computational overhead, we partition the experimental space into four operational axes. **Structural Decoding Regimes** encompass constrained decoding backends (`xgrammar`, `Outlines`, EBNF CFGs) alongside various output formats (JSON, YAML, XML, DST). Furthermore, **Serving-Layer Constraints** icluding memory fragmentation (`block_size`), decode budgeting (`max_new_tokens`), and prefill limits (`max_num_batched_tokens`, `max_model_len`). Finally, **Throughput Dynamics (Concurrency)** Batch size (`batch_size`), vLLM scheduler capacity (`max_num_seqs`). Moreover, **Task Specification** covers prompt taxonomy and few-shot scaling; as it is secondary to our systems-level focus, more information can be found in A.4. For the full list of the evaluated parameters, see Table 3.

## 4 Methodology

The detailed benchmark overview table presented in the Appendix A.2

### 4.1 Experimental Setup

We conduct our evaluations across NVIDIA A100 (40GB) and H100 (96GB) GPUs to observe energy scaling across compute generations. For generative models, we propose standardising IE energy benchmarking on the vLLM engine (v0.8.4), thereby enabling broader compatibility across GPU

generations (Kwon et al., 2023), while vLLM's PagedAttention mechanism (Niu et al., 2025b) delivers state-of-the-art memory efficiency and high throughput. Crucially, vLLM demonstrates exceptional energy efficiency during the decoding phase (Niu et al., 2025b), which is crucial for IE tasks where the generation stage is dominant. The full description of the experimental setup is presented in A.7.

### 4.2 Down-stream Tasks and Datasets

To map the energy-accuracy frontier across varying structural demands, our benchmark evaluates three Information Extraction tasks of escalating complexity: NER, EE, RE.

**NER** serves as a foundational token-level baseline, isolating the fundamental energy costs of structural formatting with minimal reasoning overhead. This evaluation primarily centres on the XTREME PAN-X dataset for broad multilingual analysis, with secondary experiments on MasakhaNER to assess low-resource generalisation. Increasing in complexity, **EE** (RAMS 1.0c) shifts the workload to multi-sentence semantic reasoning, requiring models to identify event triggers and map entity spans to 139 distinct roles. Finally, **RE** (DocRED) exhibits the highest structural complexity, requiring document-level, multi-hop reasoning to track entities and classify relationships into 96 Wikidata types across long contexts. By tracking this progression from simple tagging to complex document reasoning, the benchmark systematically isolates how task difficulty dynamically alters the energy costs of both classical encoders and generative decoders.

### 4.3 Output formats and decoding strategies.

We evaluate three serialisation formats and four decoding strategies:

**Serialisation formats:** We consider four formats: **JSON** (key–value pairs with strict syntax, serving as the de facto standard for structured outputs); **YAML** (an indentation-based, human-readable format relying on whitespace rather than brackets); **XML** (a tag-based markup format providing unambiguous nesting at the cost of higher verbosity); and **DST** (a compact, line-oriented domain-specific template designed for information extraction with minimal punctuation overhead).

**Decoding strategies:** We compare four generation approaches: **Unconstrained** (baseline, free generation without structural guarantees); **xgrammar** (compiles a JSON Schema into a dynamic token mask for single-pass structural validity; **EBNF** (uses hand-written Extended Backus - Naur Form grammars for fine-grained control over allowed sequences); and **Outlines** (enforces regex or grammar using a token-level finite-state machine, offering high generality at the cost of FSM construction overhead).

### 4.4 Choise of Models

We evaluate several general-purpose instruction-tuned LLMs spanning two families and a four parameter range: the Gemma 3 family (4B, 12B, and 27B) (Team et al., 2025) to isolate intra-family scaling effects on performance and energy, Llama 3.3 70B (Grattafiori et al., 2024) as a high-parameter inter-family reference point. For more details, see Appendix A.3. For encoder approaches across the three tasks, we compare against fully supervised architectures: DREEAM (Ma et al., 2023) for document-level RE (guided self-attention), the Multi-Sentence Argument Linking framework (Ebner et al., 2020) for EE (cross-sentence dependencies), and a projection-based pipeline (Politov et al., 2025) for NER.

### 4.5 Baseline

To establish a realistic point of comparison, we define a **naïve production baseline** representing a standard vLLM deployment for high-throughput IE. We employ general engine defaultssuch as `max_model_len=1024` and an HTTP `batch_size=1024` that accommodate all task inputs but remain untuned to the specific structural demands of IE. Furthermore, we enable standard throughput optimisations (prefix caching and chunked prefill) while leaving the Key-Value-cache unquantised (`auto`) on `bfloat16` weights to ensure a clean thermodynamic reference. This setup provides a well-defined, unoptimised anchor against which all parameter ablations are measured. A few primary serving constraints are highlighted throughout our analysis; the exhaustive baseline configuration is detailed in Appendix A.6.

## 5 Empirical Evaluation

### 5.1 Deployment Regimes: Task-Specific Supervision and Zero-Shot Flexibility

Direct comparisons between fully supervised encoder systems and zero-shot LLMs are inherently asymmetric. The evaluation situates encoder mod-

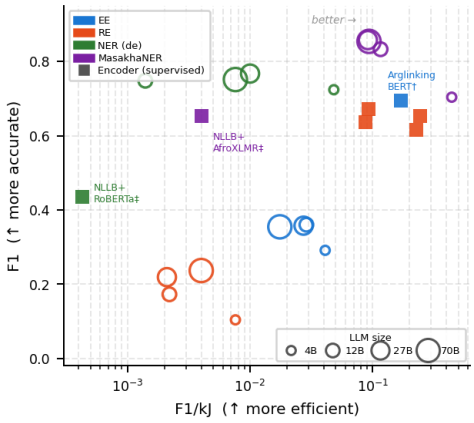


Figure 2: Performance of Encoder vs. Decoder Architectures Across Diverse Tasks. Open circles denote decoder LLMs at their best configuration per task and model size; filled squares denote supervised encoder baselines. (H100 config) Top-right is optimal.

els and LLMs within different deployment regimes rather than architectures. This offers a practical perspective on when practitioners should still prefer supervised encoders over zero-shot generative decoders. Supervised encoders represent the efficiency upper bound when task-specific supervision and dedicated pipeline engineering are available. In contrast, generative LLMs become operationally attractive when supervision, language coverage, or engineering resources are limited.

To map these regimes, we establish supervised baselines across all three tasks. For RE, we evaluate DREEAM (Ma et al., 2023); for EE, Multi-Sentence Argument Linking (Ebner et al., 2020). Although Arglinking on gold spans is not directly comparable to the end-to-end span detection and role assignment required of LLMs, we include it to define the architectural upper bound for efficiency ($F_1$/kJ). All other metrics use identical evaluation protocols. For NER, we evaluate a standard translation-plus-projection pipeline (Politov et al., 2025) utilising NLLB-200 and XLM-R-Large. This allows us to contrast zero-shot flexibility against a classical multi-stage pipeline. We test this on German (XTREME), a high-resource control where supervised encoders establish the upper bound, and on Luganda (MasakhaNER), a low-resource African language. Luganda explicitly exposes the computational bottleneck in classical pipelines included a translation step.

As visualised in Figure 2, classical encoders establish the efficiency upper bound for monolingual inference efficiency when task-specific supervision is available. For document-level RE, DREEAM-BERT achieves 0.242 F1/kJ, substantially outperforming the most accurate tuned LLM (LLaMA-3.3-70B at 0.004 F1/kJ). Although the DREEAM-RoBERTa backbone attains a marginally higher F1 score (0.671 vs 0.653), BERT consumes less than half the energy (2.69 kJ vs 7.20 kJ), making it the Pareto-optimal choice. Similarly, for Event Extraction (EE), Arglinking-BERT reaches 0.170 F1/kJ, whereas the most efficient LLM configuration (Gemma-3-12B) achieves 0.091 F1/kJ at substantially lower accuracy. These results indicate that, for conventional structured prediction tasks, the flexibility of pipeline-free LLM inference incurs a significant energy cost.

In cross-lingual and low-resource settings, the efficiency trade-off shifts as classical approaches rely on energy-intensive translation pipelines. Generative decoders instead operate without explicit intermediate translation stages. On MasakhaNER (Luganda), Gemma-3-4B bypasses the translation bottleneck, achieving 0.704 F1 at only 1.68 kJ, corresponding to 0.418 F1/kJ, a very high efficiency observed across all tasks in this study.

This dynamic holds for higher-resource Named Entity Recognition (NER) as well. On the German XTREME benchmark, Gemma-3-27B achieves peak accuracy ($F_1 = 0.767$), but the smaller Gemma-3-4B provides a 14× energy reduction (5.15 kJ, 0.127 F1/kJ) while retaining an F1 of 0.655. Ultimately, while supervised encoders are the optimal deployment choice for high-resource, monolithic extraction, compact instruction-tuned decoders represent the most sustainable strategy for flexible, cross-lingual environments.

**Motivations for Zero-Shot Generative Extraction** When is LLM deployment justified? As Figure 2 illustrates, single-pass encoders define the efficiency upper bound when task-specific supervision is abundant. However, encoders lack zero-shot flexibility. In cross-lingual or low-resource settings, generative LLMs bypass energy-intensive translation pipelines, executing extraction in a single pass. Consequently, these architectures occupy distinct deployment regimes: LLMs provide a necessary, pipeline-free alternative when task-specific supervision or engineering resources are unavailable.

### 5.2 Decoding Regimes and Formats

In addressing how output format and constrained decoding affect the energy-accuracy trade-off, our

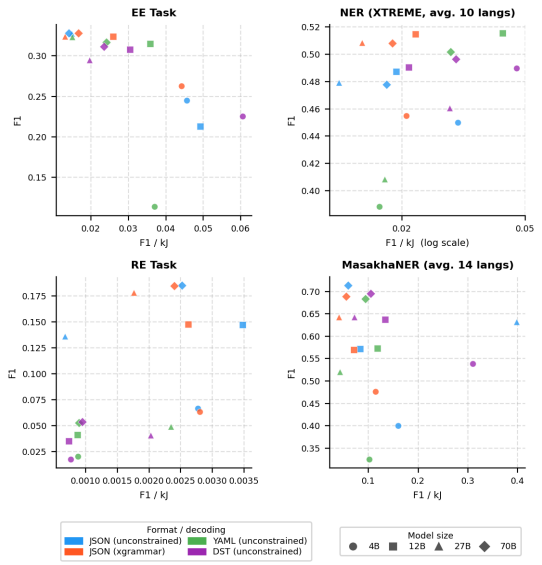


Figure 3: F1 vs. F1/kJ across four IE tasks, model scales, and output formats. NER: XTREME averages 10 languages; MasakhaNER averages 14 languages. Each point is one model-format configuration averaged across prompts (H100 config).

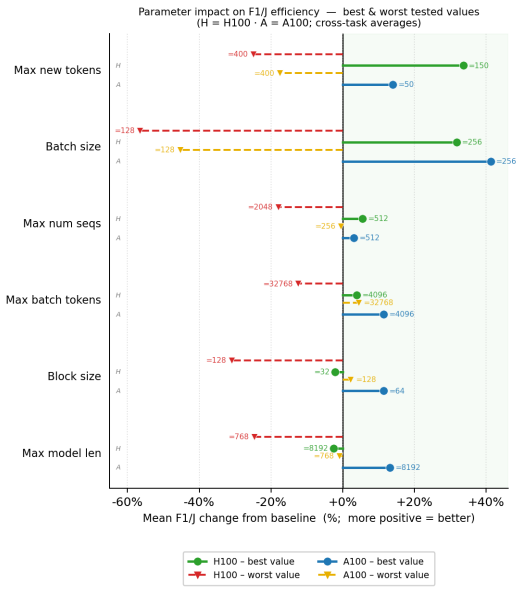


Figure 4: Cross-architectural analysis of server-side parameter variations

evaluations demonstrate that inference costs and predictive quality are dictated by the complex interaction among task rigidity, model scale, and decoding format, see Figure 3.

Constrained decoding imposes a dynamic "computational tax". In NER, constrained JSON (`xgrammar`) degrades smaller models ($-16.8\%$ F1 on 12B) but boosts larger ones (27B, 70B). Conversely, highly rigid tasks like Event Extraction (EE) benefit uniformly across all scales (up to $+74.0\%$ F1 on 12B). Formatting also strictly dictates energy consumption: JSON+`xgrammar` incurs moderate overhead (≈90% on 4B), whereas FSM-heavy methods like DST+`Outlines` or XML+`xgrammar` are energy-prohibitive, consuming up to $38\times$ baseline energy. Furthermore, accuracy is highly format-sensitive. While Domain-Specific Text (DST) optimises NER, Relation Extraction (RE) strictly requires JSON; adopting alternatives such as YAML results in severe F1 degradation (up to $-69.7\%$).

When balancing accuracy, energy, and deterministic parsing, Gemma-3-12B with JSON+`xgrammar` is the broadly applicable baseline that guarantees schema adherence with bounded overhead. For cost-agnostic peak accuracy, deploy Llama-3.3-70B with JSON+`xgrammar` for NER, unconstrained JSON for RE, and 12B JSON+`xgrammar` for EE. Conversely, strictly energy-constrained deployments should default to unconstrained JSON on 4B models.

For multilingual deployments below the 27B scale, DST+EBNF is the most robust default, as `xgrammar` masking degrades small-model performance on non-Latin scripts. For rapid prototyping, unconstrained DST provides structural guidance and consistent F1 gains at zero computational penalty.

In summary, output formats and constrained decoding are not mere implementation details, but primary drivers of the energy-accuracy trade-off. Sustainable generative IE requires deliberately aligning these mechanisms with the target task's inherent structural rigidity.

### 5.3 Influence of Serving Parameters on Energy Efficiency

To determine how much serving-layer parameters affect the efficiency frontier, we conducted a comprehensive ablation study across NVIDIA A100 and H100 GPUs (details in A.2). Our findings in Figure 4 confirm that default inference engine settings are rarely calibrated for structured IE. Ultimately, achieving optimal efficiency requires precisely aligning a strict hierarchy of serving constraints with the target task's specific complexity.

**Critical Parameters: Decode Limits and Batch Sizing** Driving efficiency swings of $\pm 50\%$ $F_1$/kJ, these parameters demand aggressive tuning. For NER, unconstrained generation severely

| Core Finding | Empirical Evidence | Practical Implication |
|---|---|---|
| **Encoders set efficiency bounds.** | DREEAM-BERT ($0.242F_1$/kJ) on RE vastly outperforms tuned LLMs ($0.004F_1$/kJ). | Prefer classical encoders for supervised tasks; use LLMs for zero-shot or cross-lingual needs. |
| **Output format determines accuracy.** | RE strictly requires JSON (up to $-69.7\%$ F1 drop without it); simple NER optimises best under DST. | Treat formatting as an architectural variable; align verbosity with task rigidity. |
| **Constrained decoding imposes a tax.** | FSMs (`Outlines`) inflate energy up to $38\times$. `xgrammar` boosts rigid tasks but hurts small NER models. | Benchmark schema enforcement. Default to JSON+`xgrammar`, avoiding heavy FSMs if energy-bound. |
| **Serving limits shift the frontier.** | The volatile triad (`batch_size`, `max_new_tokens`, `max_num_batched`) alters efficiency by $\pm 50\%$. | Aggressively tune decode budgets to prevent hallucinations; mandate reporting of serving parameters. |
| **Scale does not guarantee efficiency.** | On German NER, Gemma-3-4B offers a $14\times$ energy reduction over 27B with competitive utility. | Exhaust pipeline tuning (formats, limits) on compact 4B–12B models before scaling capacity. |

Table 1: Main findings and practical implications for deploying energy-efficient generative IE pipelines.

wastes energy: verbose chain-of-thought prompts inflate costs by 50% (9.6 vs. 6.4 J/TP), making tight decode caps (50–100 `max_new_tokens`) critical to prevent costly hallucinations. Conversely, Relation Extraction (RE) requires 200 tokens. For the client batch size (`batch_size`), moderate HTTP batches (256) alleviate memory bandwidth contention for EE and RE (yielding up to $+50.4\%$ $F_1$/kJ). Crucially, dropping the batch size too far (64-128) breaks the chunked-prefill scheduler, destroying thermodynamic efficiency across all tasks by up to $46.1\%$.

**Task-Specific Parameters: Memory and Prefill Allocation** These limits offer significant gains but require structural alignment. Scaling prefill limits (`max_num_batched_tokens`) to 16,384 uniquely optimises dense reasoning batches for prompt-heavy RE and EE tasks (up to $+44.0\%$ $F_1$/kJ). Similarly, increasing memory fragmentation blocks (`block_size`) to 32 or 64 significantly improves long-context EE and RE, but overextending it to 128 unnecessarily increases energy costs for short-span NER tasks.

**Universal Parameters: Scheduler Capacity and Context Truncation** These parameters provide universally safe, low-variance optimisations without the volatility of the primary variables. Reducing scheduler capacity (`max_num_seqs`) from the default 1024 to 512 consistently saves energy across all tasks and hardware. It eliminates the hidden energy tax of over-saturating the scheduler without sacrificing throughput, yielding a stable $+6.2\%$ $F_1$/kJ gain. Finally, safely truncating the maximum context window (`max_model_len`) to match actual document lengths (4096 or 8192) recovers predictive accuracy while securing moderate efficiency improvements (up to $+13.2\%$).

**Influence of parameters:** Our comprehensive ablation study reveals that `batch_size`, `max_new_tokens`, and `max_num_batched_tokens` are the most influential serving parameters governing the thermodynamic efficiency of LLMs in IE. These three highly volatile variables trigger massive performance swings, altering efficiency ($F_1$/kJ) by up to $\pm 50\%$. Specifically, `max_new_tokens` strictly bounds the decoding budget to prevent energy-intensive hallucinations in short-span tasks such as NER. Concurrently, tuning the client `batch_size` is critical for avoiding memory bandwidth contention and maintaining scheduler stability, while scaling `max_num_batched_tokens` specifically optimises dense reasoning batches for prompt-heavy tasks. Consequently, correctly aligning this triad of primary parameters with the structural demands of the target task is the first critical step in deploying efficient, frontier-sustainable generative IE pipelines.

### 5.4 Main Findings

Our comprehensive evaluation demonstrates that the energy efficiency of LLM-based IE cannot be predicted by model scale alone. Instead, achieving optimal F1 and energy trade-offs requires precise alignment of prompt design, decoding structures, and serving parameters. We synthesise our core findings in Table 1.

### 5.5 Scenario-Driven Configuration Guidelines

To operationalise these trade-offs and provide actionable setup guidance, we categorise the optimal

parameter configurations and their empirical gains relative to the unoptimised baseline across four distinct deployment scenarios:

**Scenario A: Peak Accuracy.** Settings: Maximum representational scale (27B/70B), extended context limits (`max_model_len`=8192), and strict FSM structural enforcers (e.g., DST+`Outlines`). **Outcome vs. Baseline:** Achieves the absolute peak predictive utility (e.g., $+16.4\%$ F1 on NER). However, this setup incurs a massive thermodynamic penalty. Compared to the unconstrained baseline, complex FSM pre-computation inflates energy consumption to pathological levels (up to $38\times$), making this viable only for offline, high-stakes extraction.

**Scenario B: Maximum Tuning Gain.** Settings: Aggressive alignment of transport parameters, specifically reducing HTTP `batch_size` to 256 and scaling `max_num_batched_tokens` to 16,384 for dense reasoning batches. **Outcome vs. Baseline:** Demonstrates the absolute value of parameter tuning. By alleviating memory bandwidth contention and optimising chunked prefill, this setup yields massive efficiency surgesboosting $F_1/\mathrm{J}$ by up to $+50.4\%$ over the default baseline for prompt-heavy tasks like Event and Relation Extraction.

**Scenario C: Baseline Efficiency.** Settings: Default vLLM deployment (`batch_size`=1024, `max_num_seqs`=1024) paired with compact models (4B) and unconstrained JSON formatting (no grammar engines). **Outcome vs Baseline:** The required setup when engineering overhead is unavailable, and energy is strictly constrained. While deploying the 4B model yields a native $14\times$ energy reduction compared to 27B/70B baselines, using unconstrained generation on small models risks catastrophic accuracy collapse (e.g., $-69.7\%$ F1 on RE) due to format deviations.

**Scenario D: Pareto-Optimal Default.** Settings: Mid-scale models (Gemma-3-12B) combined with lightweight JSON+`xgrammar` constraints, capped decode budgets (`max_new_tokens`=100), and a reduced scheduler capacity (`max_num_seqs`=512). **Outcome vs Baseline:** The most robust configuration under the tested setup. Compared to the unoptimised baseline, this configuration prevents hallucinated token generation and eliminates hidden scheduler taxes. It recovers at least 80 % of the peak F1 while yielding transformative efficiency gainssuch as a $+74.0\%$ F1 surge in EE without the catastrophic energy spikes of Scenario A.

### 5.6 Discussion

Our findings indicate that energy-aware evaluation for generative Information Extraction must move beyond reporting model size, GPU type, or raw Energy-per-Token statistics. Because structured outputs and constrained decoding fundamentally alter inference behaviour, energy efficiency in generative IE is largely a systems-level property rather than a direct consequence of parameter count. As a result, two deployments of the same model can exhibit substantially different energy profiles depending on the serving configuration. For reproducible and meaningful benchmarking, we recommend reporting the structural properties of the extraction pipeline alongside predictive metrics, including the output format, the constrained decoding strategy, the parser enforcement mechanism, the decode budget, and the serving configuration. These operational details strongly influence both inference behaviour and energy consumption, and without them, comparisons across models and deployment environments remain difficult to interpret reliably.

## 6 Conclusion

As Information Extraction (IE) rapidly transitions toward generative architectures, understanding the true operational costs of inference is critical. In this work, we introduced the first dedicated benchmark for energy-efficient IE, moving beyond standard token-throughput metrics to utilise task-aligned thermodynamic indicators (such as $J/TP$ and $F_1/\mathrm{kJ}$).

Crucially, our evaluations demonstrate that the energy efficiency of LLM-based IE cannot be predicted by model size or any single parameter alone. Instead, efficiency is largely determined by the complex interplay among architectural paradigms, structural decoding formats, and serving-layer dynamics. By systematically mapping the influence of these specific parameters on the energy-accuracy trade-off, this benchmark provides actionable, scenario-driven deployment guidelines. We explicitly outline exactly what configuration settings to use and when, enabling practitioners to properly set up their pipelines to achieve peak predictive accuracy, maximise energy efficiency, or establish a targeted Pareto-optimal balance. Ultimately, this work equips the NLP community with the empirical foundation necessary to deploy highly capable, yet thermodynamically sustainable, Information Extraction systems.

## 7 Ethical Impact

The rapid adoption of Large Language Models (LLMs) for Information Extraction (IE) raises an ethical question that cannot be answered by accuracy alone: when is the marginal improvement in extraction quality worth the additional energy and environmental cost? Our results show that generative IE systems can impose substantial energy overhead through model scale, constrained decoding, serialisation format, and serving-layer choices. This supports the broader Green AI argument of Schwartz et al. (2020), who argue that efficiency should be treated as a first-class evaluation criterion alongside accuracy, especially because compute-intensive AI can exclude researchers and practitioners without access to large infrastructure.

However, the ethical response is not simply to always choose the smallest or cheapest model. In high-stakes domains, such as medical, legal, financial, or humanitarian extraction, additional energy may be justified if it prevents harmful errors; similarly, in low-resource and cross-lingual settings, compact zero-shot LLMs may be ethically preferable when they replace brittle multi-stage translation pipelines and expand access to languages for which supervised IE systems are unavailable. We therefore argue for a principle of proportional deployment: practitioners should select the lowest-energy configuration that meets the task's reliability threshold, and justify any move to larger models or stricter decoding constraints by measurable gains in useful extractions. Task-aware metrics such as F1-Energy Ratio ($F1/kJ$) and Energy per True Positive make this trade-off explicit, while recent LLM inference research further shows that optimisation choices alone can reduce total energy use by up to 73% in some settings (Fernandez et al., 2025).

However, optimising the energy efficiency of individual extractions is merely a prerequisite for sustainable deployment, rather than a complete solution. A central ethical risk is the rebound effect: making IE cheaper per document may encourage organisations to process more documents, run more retries, deploy extraction in lower-value contexts, or normalise large-scale automated analysis that would previously have been considered too costly. In this case, a system may become more efficient per extraction while increasing total computation and environmental burden. This is especially important because environmental impact is broader than runtime GPU energy: Luccioni et al. (2023) shows that carbon accounting changes substantially when equipment manufacturing and other life-cycle processes are included, and recent environmental AI research warns that energy and $CO_2$ metrics often underrepresent systemic effects such as water use, material extraction, infrastructure expansion, and demand growth (Carmeno et al., 2026). Consequently, energy-efficient IE benchmarks should not only identify better configurations, but also support more restrained deployment decisions: who benefits from the extraction, whether the task is socially meaningful, how often the system will be run, and whether efficiency gains are being used to reduce total impact or simply to scale automation further.

## 8 Limitations

While our benchmark captures critical efficiency dynamics, its scope is limited to specific extraction tasks (NER, RE, EE) and models (Gemma-3, Llama-3.3, used per their licenses), meaning trends may not generalise to alternative architectures or retrieval-augmented pipelines. To ensure reproducible comparisons, our measurements strictly isolate the energy used for active GPU inference. By excluding model initialisation, system-level overhead (e.g., CPU, cooling), and grid variables, we deliberately avoid broader carbon-footprint claims. Additionally, our Hyperparameter Optimisation (HPO) trial was exploratory rather than exhaustive. Finally, our batching and scheduling findings are inherently tied to the vLLM stack and NVIDIA hardware and may not transfer to other frameworks. Notably, the H100 served as our primary evaluation platform; establishing a direct architectural comparison with the A100 falls outside the core objectives of this study.

## Acknowledgments

## References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. 2026. Gepa: Reflective prompt evolution can outperform reinforcement learning. *Preprint*, arXiv:2507.19457.

Mufakir Qamar Ansari and Mudabir Qamar Ansari. 2025. Racing to idle: Energy efficiency of matrix multiplication on heterogeneous cpu and gpu architectures. *Preprint*, arXiv:2507.20063.

Pierre Bourhis, Juan L. Reutter, and Domagoj Vrgo. 2020. Json: Data model and query languages. *Information Systems*, 89:101478.

Noemi Luna Carmeno, Tiago Domingos, and Daniel W. O'Neill. 2026. The impacts of artificial intelligence on environmental sustainability and human well-being. *Preprint*, arXiv:2602.24091.

Krishna Teja Chitty-Venkata, Siddhisanket Raskar, Bharat Kale, Farah Ferdaus, Aditya Tanikanti, Ken Raffenetti, Valerie Taylor, Murali Emani, and Venkatram Vishwanath. 2025. Llm-inference-bench: Inference benchmarking of large language models on ai accelerators. In *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '24, page 13621379. IEEE Press.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2024. Xgrammar: Flexible and efficient structured generation engine for large language models. *ArXiv*, abs/2411.15100.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8057–8077, Online. Association for Computational Linguistics.

Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. 2025. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32556–32569, Vienna, Austria. Association for Computational Linguistics.

Iker García-Ferrero, Rodrigo Agerri, and German Rigau. 2022. Model and data transfer for cross-lingual sequence labelling in zero-resource settings. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6403–6416, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: a massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Hongzhen Huang, Kunming Zhang, Hanlong Liao, Kui Wu, and Guoming Tang. 2025. Aimeter: Measuring, analyzing, and visualizing energy and carbon footprint of ai workloads. *Preprint*, arXiv:2506.20535.

Lena Jurkschat, Preetam Gattogi, Sahar Vahdati, and Jens Lehmann. 2025. Bali - a benchmark for accelerated language model inference. *IEEE Access*, 13:98976–98989.

Saicharan Kolluru. 2025. Comparative analysis of large language model inference serving systems: A performance study of vllm and huggingface tgi. *Preprint*, arXiv:2511.17593.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611626, New York, NY, USA. Association for Computing Machinery.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. 2024. Sprout: Green generative AI with carbon-efficient LLM inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21799–21813, Miami, Florida, USA. Association for Computational Linguistics.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. 2023. Estimating the carbon footprint of bloom, a 176b parameter language model. *J. Mach. Learn. Res.*, 24(1).

Youmi Ma, An Wang, and Naoaki Okazaki. 2023. DREEAM: Guiding attention with evidence for improving document-level relation extraction. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1971–1983, Dubrovnik, Croatia. Association for Computational Linguistics.

Haosi Mo, Xinyu Ma, Xuebo Liu, Derek F. Wong, Yu Li, Jie Liu, and Min Zhang. 2025. CDT: A comprehensive capability framework for large language models across cognition, domain, and task. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3715–3734, Suzhou, China. Association for Computational Linguistics.

Shiwen Ni, Guhong Chen, Shuaimin Li, Xuanang Chen, Siyi Li, Bingli Wang, Qiyao Wang, Xingjian Wang, Yifan Zhang, Liyang Fan, Chengming Li,

Ruifeng Xu, Le Sun, and Min Yang. 2025. A survey on large language model benchmarks. *Preprint*, arXiv:2508.15361.

Chenxu Niu, Wei Zhang, Yongjian Zhao, and Yong Chen. 2025a. Energy efficient or exhaustive? benchmarking power consumption of llm inference engines. *SIGENERGY Energy Inform. Rev.*, 5(2):5662.

Chenxu Niu, Wei Zhang, Yongjian Zhao, and Yong Chen. 2025b. Energy efficient or exhaustive? benchmarking power consumption of llm inference engines. *SIGENERGY Energy Inform. Rev.*, 5(2):5662.

David Pape, Jonathan Evertz, and Lea Schönherr. 2026. The silent hyperparameter: Quantifying the impact of inference backends on llm reproducibility. *Preprint*, arXiv:2605.19537.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Andrei Politov, Oleh Shkalikov, Rene Jäkel, and Michael Färber. 2025. Revisiting projection-based data transfer for cross-lingual named entity recognition in low-resource languages. In *Proceedings of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025)*, pages 499–507, Tallinn, Estonia. University of Tartu Library.

Alberto Poncelas, Maksim Tkachenko, and Ohnmar Htun. 2023. Sakura at SemEval-2023 task 2: Data augmentation via translation. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 1718–1722, Toronto, Canada. Association for Computational Linguistics.

Zhiwen Ruan, Yixia Li, Yefeng Liu, Yun Chen, Weihua Luo, Peng Li, Yang Liu, and Guanhua Chen. 2025. G2: Guided generation for enhanced output diversity in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14116–14134, Suzhou, China. Association for Computational Linguistics.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. GoLLIE: Annotation guidelines improve zero-shot information-extraction. In *The Twelfth International Conference on Learning Representations*.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM*, 63(12):5463.

Zhengyuan Shen, Darren Yow-Bang Wang, Soumya Smruti Mishra, Zhichao Xu, Yifei Teng, and Haibo Ding. 2025. SLOT: Structuring the output of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 472–491, Suzhou (China). Association for Computational Linguistics.

Lola Solovyeva and Fernando Castor. 2026. Towards green ai: Decoding the energy of llm inference in software development. *Preprint*, arXiv:2602.05712.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Christian Wassermann, Tobias Dollenbacher, Christian Terboven, and Matthias S. Müller. 2026. Leveraging nvml gpm for nvidia gpu monitoring. In *Proceedings of the Supercomputing Asia and International Conference on High Performance Computing in Asia Pacific Region Workshops*, SCA/HPCAsiaWS '26, page 5968, New York, NY, USA. Association for Computing Machinery.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *Preprint*, arXiv:2307.09702.

Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin Lee, and 6 others. 2022. Sustainable AI: Environmental Implications, Challenges and Opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813.

Hui Wu, Xiaoyang Wang, and Zhong Fan. 2025. Addressing the sustainable ai trilemma: a case study on llm agents and rag. *ArXiv*, abs/2501.08262.

Zeyu Yang, Karel Adamek, and Wesley Armour. 2024. Part-time power measurements: nvidia-smi's lack of attention. *Preprint*, arXiv:2312.02741.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics.

Aaron Yu, Jochen Weile, and Mélanie Courtot. 2026. Benchmarking llm-based information extraction tools for medical documents. *medRxiv*.

Zikang Zhang, Wangjie You, Tianci Wu, Xinrui Wang, Juntao Li, and Min Zhang. 2025. A survey of generative information extraction. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4840–4870, Abu Dhabi, UAE. Association for Computational Linguistics.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. 2024. A survey on efficient inference for large language models. *Preprint*, arXiv:2404.14294.

# A Appendix

## A.1 Software Environment

All experiments were executed under the following software stack.

- **OS & Workload manager:** SLURM 23.11.8; Operating system: Rocky Linux 9.6
- **Container runtime:** Apptainer 1.4.5 (formerly Singularity); The vLLM serving stack was packaged as a single `.sif` image built from a custom Dockerfile (`nvidia/cuda:12.8` base)
- **CUDA:** 12.8 (host driver); CUDA 11.7 inside the inference container
- **Inference server:** vLLM 0.8.4, served via HTTP with OpenAI-compatible API
- **Python:** 3.10 (container) / 3.11 (evaluation environment); key packages: `transformers`, `seqeval`, `aiohttp`, `mlflow`, `prometheus_client`
- **Energy monitoring:** NVIDIA DCGM Exporter via Prometheus (scrape interval 100 ms); custom NVML poller at 10 Hz for per-job GPU power traces
- **Module system:** Lmod 9.2

## A.2 Benchmark Overview

To establish a clear foundation for our empirical analysis, Table 2 provides a comprehensive architectural overview of the energy-aware generative Information Extraction benchmark. It explicitly outlines the core dimensions of our evaluation framework, systematically categorising the targeted downstream tasks, model families, and the specific constrained decoding regimes applied.

| Dimension | Benchmark Components |
|---|---|
| **Tasks** | Named Entity Recognition (NER), Event Extraction (EE), Relation Extraction (RE) |
| **Datasets** | XTREME PAN-X (10 langs), MasakhaNER 2.0 (14 langs), RAMS 1.0c (En), DocRED (En) |
| **Languages** | **XTREME** (91 000 test sentences): Arabic, German, English, Spanish, French, Japanese, Korean, Russian, Chinese (10 000 each), Hindi (1 000)<br>**MasakhaNER** (22 210 test sentences): Ewe (1 001), Hausa (1 633), Igbo (2 181), Kinyarwanda (2 235), Luganda (1 412), Chichewa (1 785), Nigerian Pidgin (1 294), Kiswahili (1 883), Setswana (996), Akan/Twi (1 211), Wolof (1 312), isiXhosa (1 633), Yorùbá (1 964), isiZulu (1 670)<br>**RAMS 1.0c:** 871 test events (English)<br>**DocRED:** 1 000 test documents (English) |
| **Volume** | 3,488 planned configurations (model × task × language × format × prompt); each configuration repeated $N$=3 times → 9,291 total GPU inference runs |
| **Model Families** | **Decoders:** Gemma-3 (4B, 12B, 27B), Llama-3.3 (70B), GoLLIE<br>**Encoders:** DREEAM, Argument Linking, NLLB+XLM-R |
| **Output Formats** | JSON, YAML, XML, DST (Domain-Specific Text) |
| **Decoding** | Unconstrained, `xgrammar`, EBNF, `Outlines` |
| **Serving Limits** | `batch_size`, `max_new_tokens`, `max_num_batched_tokens`, `block_size`, `max_num_seqs`, `max_model_len` |
| **Hardware** | **Node type A:** 8× NVIDIA A100-SXM4 (40 GiB HBM2), 2× AMD EPYC 7352 (24 cores, 2.3 GHz), 1 TiB RAM<br>**Node type B:** 4× NVIDIA H100-SXM5 (94 GiB HBM2e), 2× AMD EPYC 9334 (32 cores, 2.7 GHz), 768 GiB RAM<br>SLURM cluster; vLLM v0.8.4 |
| **Protocol** | $N = 3$ runs; active GPU inference window isolated via NVIDIA DCGM |
| **Metrics** | **Predictive:** Micro-averaged F1, Precision, Recall<br>**Thermodynamic:** F1-Energy Ratio ($F_1$/kJ), Energy per True Positive ($J/TP$), Energy per Extraction ($E_{\text{extraction}}$) |

Table 2: Overview of the IE benchmark, presented in the paper

## A.3 Model deployment

All generative models used in this study (Gemma-3, Llama-3.3) were accessed in accordance with their respective community licenses and terms of use. We strictly adhered to the usage policies set forth by the model providers, ensuring that all outputs were used solely for non-commercial research purposes and that no prohibited content generation

| Parameter | Evaluated Variations |
|---|---|
| **Task Specification** | |
| Prompt Styles | P1–P9 (Across all NER models) |
| Output Formats | JSON<br>DST<br>XML<br>YAML |
| Decoding Strategies | Unconstrained<br>JSON + `xgrammar`<br>XML + `xgrammar`<br>DST + `ebnf`<br>DST + `Outlines` |
| **Concurrency Limits** | |
| `max_concurrency` | 256, 512, 1024 |
| `batch_size` | 128, 512, 1024 |
| `max_num_seqs` | 512, 1024, 2048 |
| **Context & Budgets** | |
| `max_new_tokens` | 50, 100, 150, 200, 250, 300 |
| `max_model_len` | 768, 1024, 2048 |
| `max_num_batched_tokens` | 8192, 16384, 32768 |
| **vLLM Engine Flags** | |
| `block_size` | 16, 32, 64, 128 |
| `kv_cache_dtype` | `auto`, `fp8` |
| `dtype` | `bfloat16` |
| `enable_prefix_caching` | `True`, `False` |
| `enable_chunked_prefill` | `True`, `False` |
| `calculate_kv_scales` | `True`, `False` |
| **Hardware Topology** | |
| `tensor_parallel_size` | Scaled to physical GPUs |

Table 3: Comprehensive taxonomy of all architectural, structural, and serving-layer parameters evaluated across the benchmark sweeps.

| Parameter | Description |
|---|---|
| *Output & Decoding* | |
| Prompt style | Instruction phrasing variant (P1–P9) |
| Output format | JSON, DST, YAML, XML |
| Decoding | Unconstrained or grammar constrained (`xgrammar`, `ebnf`, Outlines) |
| *Concurrency* | |
| `max_concurrency` | Max parallel HTTP requests |
| `batch_size` | Client-side request batch size |
| `max_num_seqs` | vLLM in-flight sequence cap |
| *Context & Token Budgets* | |
| `max_new_tokens` | Per-request generation budget |
| `max_model_len` | Maximum prompt + response length |
| `max_num_batched_tokens` | Prefill window size |
| *Engine Flags* | |
| `block_size` | KV cache page size (tokens) |
| `kv_cache_dtype` | KV cache precision (`fp8`/`auto`) |
| `enable_prefix_caching` | Reuse KV cache across shared prefixes |
| `enable_chunked_prefill` | Split long prefills across steps |
| `calculate_kv_scales` | Dynamic FP8 KV scale computation |
| `tensor_parallel_size` | GPU parallelism degree |

Table 4: Serving and inference parameters varied in the benchmark sweep.

was performed. Hardware allocation was scaled according to model parameters and architectural requirements. On the NVIDIA H100 infrastructure, the Gemma model family was deployed on a single GPU, whereas the Llama models were deployed on a 4-GPU configuration. Conversely, deployments on the NVIDIA A100 hardware scaled incrementally: the Gemma 4B and 12B variants were hosted on a single GPU, the Gemma 27B model required a 2-GPU setup, and the 70B model was distributed across eight A100 GPUs. Moreover, we compared Gollie (Sainz et al., 2024), fine-tuned specifically for information extraction, with other LLMs on a configuration with A100 GPUs (1 GPU).

### A.4 Prompt Configurations

To systematically isolate the impact of prompt design on both performance and energy consumption, we evaluated nine prompt variants (P1–P9) for the NER task and eight variants (P1–P8) for EE and RE. The configurations span a spectrum from minimal zero-shot instructions (P1) to structured bullet-point prompts (P3–P6), rule-based verbose templates inspired by PromptNER (P7), and chain-of-thought reasoning prompts (P8). Variants P2–P6 additionally inject language-specific few-shot examples (one to three input–output pairs) into the context. For NER only, variant P9 uses instructions produced by the DSPy GEPA (Gradient-free Ensemble Prompt Adaptation) optimiser (Agrawal et al., 2026): a GEPA run is executed per language on the XTREME-PAN-X benchmark, so the resulting optimised instructions are both task-specific and language-specific, and give results on the level of the simple P1 prompt. Short prompt overview provided in Table 5. Full prompts are presented in the GitHub repository. Unless stated otherwise, energy figures reported throughout this work are measured on a fixed baseline configuration, enabling accurate isolation of generation cost from prompt-induced variability.

### A.5 Prompt Design

Prompt design details can be found in Table 5

### A.6 Baseline Experimental Configuration

Table 6 details the complete unoptimised vLLM serving configuration used to establish the baseline energy and throughput metrics prior to the ablation study.

### A.7 Energy Measurement Pipeline

To ensure reproducible, high-fidelity measurements across isolated hardware environments, all energy telemetry was managed through a dedicated hardware-level monitoring stack. Because the foundational metric definitions and active-window boundaries, this section details the operational mechanics of the pipeline's execution on High-Performance Computing (HPC) infrastructure.

| ID | Type | Design description |
|---|---|---|
| P1 | Basic zero-shot | Minimal instruction: task + output format only; no rules or examples. |
| P2 | Basic one-shot | P1 + single annotated demonstration example. |
| P3 | Rules zero-shot | Structured bullet-point rules (verbatim, no hallucination, JSON); no example. |
| P4 | Rules one-shot | P3 rules + one annotated demonstration example. |
| P5 | Rules two-shot | P3 rules + two annotated demonstration examples. |
| P6 | Extended three-shot | Expanded rules (entity boundary constraints, cross-lingual note) + three examples. |
| P7 | PromptNER rules | PromptNER-style explicit rule list (zero-shot). |
| P8 | Chain-of-thought | Two-phase CoT: internal reasoning then final JSON; one CoT demonstration. |
| P9 | Alt-format | Non-JSON output format (`### per / org / loc` markers). |

Table 5: Prompt design summary

| Parameter | Baseline Value |
|---|---|
| **Serving-Layer Constraints** | |
| `max_model_len` | 1024 |
| `max_new_tokens` | 150 |
| `max_num_batched_tokens` | 8192 |
| `block_size` | 16 |
| **Throughput & Concurrency** | |
| `batch_size` | 1024 |
| `max_concurrency` | 256 |
| `max_num_seqs` | 1024 |
| **Engine Optimisations** | |
| `dtype` | `bfloat16` |
| `kv_cache_dtype` | `auto` |
| `enable_prefix_caching` | `True` |
| `enable_chunked_prefill` | `True` |
| `calculate_kv_scales` | `False` |
| **Hardware Topology** | |
| `tensor_parallel_size` | Equal to physical GPU count |

Table 6: Naïve Production Baseline: Default vLLM Serving Configuration

**HPC Orchestration and Strict Isolation.** To prevent thermodynamic or scheduler interference from parallel workloads, experiments were submitted via a SLURM workload manager. The orchestrator script enumerated the Cartesian product of all evaluated axes (model sizes, languages, constrained-decoding backends, and vLLM serving limits) and dispatched a single exclusive job per configuration to dedicated, fully isolated A100 or H100 compute nodes (e.g., the Capella and Alpha partitions).

**Containerised Serving and Telemetry.** To avoid host-level dependency conflicts and ensure identical execution environments across GPU architectures, the pipeline utilised a dual-container architecture via Apptainer/Singularity. The target LLM was served via vLLM within the primary container, which exposed an OpenAI-compatible REST API. Concurrently, NVIDIA's DCGM exporter (`nvidia/dcgm-exporter`) ran in a secondary container. Exposing a Prometheus-compatible HTTP endpoint, this secondary container directly accessed the GPU hardware management interface, ensuring the telemetry stream remained entirely independent of the inference server's software stack.

**Asynchronous Dispatch and Polling.** Instead of sequential processing, the inference evaluation script dispatched prompts to the vLLM server via asynchronous HTTP requests. The concurrency of these requests was strictly bounded by a configurable semaphore to match the target throughput dynamics of the given trial. Simultaneously, a lightweight background Python process polled the DCGM exporter endpoint. By appending hardware state data (including board power, VRAM utilisation, and die temperature) to a per-job registry at 100 ms intervals, the pipeline effectively captured the transient hardware spikes induced by chunked prefilling and constrained FSM compilation.

**Dynamic Alignment and Artefact Logging.** Following gold-label parsing, telemetry was dynamically aligned to the exact window of $> 0\,\%$ GPU utilisation, isolating active inference energy ($\Delta E$) from loading overhead. All resulting artefacts (latency, efficiency metrics, parsing success) were then logged to MLflow to map the Pareto frontier.

### A.8 Concurrency on Different Hardware Architectures

**Concurrency and Serving-Layer Bottlenecks.** The client-side concurrency ceiling ($C$) dictates the volume of parallel inference requests. While conventionally assumed to maximise GPU utilisation, pushing $C$ too high for fast, compact models (e.g., 4B on the large-scale XTREME NER task) severely degrades thermodynamic efficiency. In our setup, the observed performance degradation is consistent with a serving-layer bottleneck, such as connection exhaustion. Because small models resolve requests rapidly (1–3 seconds), they appear to open and close connections at a rate that outpaces

**(a) Entity Count vs. Success Rate**

| Run (B, seq) | Entities | % vs A | Status |
|---|---|---|---|
| A (512, 512) | 12,765 | 100.0% | All succeed |
| B (1024, 512) | 8,941 | 70.0% | ∼30% fails |
| C (512, 1024) | 13,243 | 103.7% | All succeed |
| D (1024, 1024) | 7,892 | 61.8% | ∼38% fails |

**(b) Run D Telemetry: Alternating Pattern**

| Cohort | Tokens | Reqs | Status |
|---|---|---|---|
| 1 | 77,842 | ∼1,024 | Full |
| 2 | 450 | ∼6 | Nearly empty |
| 3 | 77,382 | ∼1,024 | Full |
| 4 | 1,626 | ∼21 | Nearly empty |
| 5 | 77,900 | ∼1,024 | Full |
| 6 | 2,981 | ∼39 | Nearly empty |

Table 7: Concurrency diagnostics demonstrating the correlation between entity yield and HTTP timeouts (a), alongside the alternating telemetry signature indicative of TCP port exhaustion (b).

the operating system's socket recycling window (e.g., `TIME_WAIT`). This likely triggers exhaustion of the ephemeral port pool, in which a saturated network stack drops subsequent request cohorts, causing massive connection failures and wasting baseline power.

**Recall as a Systems Diagnostic.** Under such serving-layer constraints, Recall emerges as a critical diagnostic marker for server bottlenecks. Because dropped connections or network congestion discard entire outputs, Recall collapses monotonically as $C$ increases, whereas Precision remains artificially stable by only reflecting successful parses. Crucially, this bottleneck scales with hardware throughput: because the H100 processes requests faster than the A100, it exacerbates the observed connection saturation, triggering a much steeper Recall drop for the 4B model at $C$=1024 ($-21\%$ on H100 vs. $-3.7\%$ on A100).

**Latency and Scale Dependencies.** This serving-layer failure mode is bypassed entirely by larger models (e.g., 27B, 70B), where the longer per-request service time naturally prevents socket saturation. Similarly, smaller datasets (EE, RE) fail to generate enough sustained request volume to overwhelm the connection pool. Consequently, optimal concurrency is not a static hyperparameter, but a dynamic threshold strictly bounded by the intersection of model latency, dataset volume, and the host operating system's network stack capacity.

**Concurrency and TCP Exhaustion Analysis**

Evidence from entity counts demonstrates that output yield directly mirrors HTTP success rates (Table 7a). Specifically, Run D (`max_num_seqs=1024`) yielded fewer entities (61.8% of baseline) compared to Run B (`max_num_seqs=512`, yielding 70.0%). This indicates that rather than alleviating bottlenecks, pushing the vLLM scheduler to handle more concurrent requests exacerbates memory churn, scheduling overhead, and HTTP-level timeouts.

Furthermore, per-cohort telemetry from Run D reveals a distinctive alternating pattern (Table 7b). Full cohorts of approximately 1,024 requests are followed immediately by nearly empty cohorts. This signature strongly suggests TCP connection exhaustion: a saturated cohort opens numerous connections that enter a `TIME_WAIT` state. The subsequent cohort fires into a congested transport layer, causing widespread connection failures until the operating system recycles sufficient sockets.

### A.9 Comparison with GoLLIE: Fine-Tuned Specialist vs. Zero-Shot Generalist

GoLLIE (Sainz et al., 2024) is a 7B-parameter model (CodeLLaMA-7B) fine-tuned Specifically for information extraction using annotated, code-style supervision. We ran GoLLIE under the same serving infrastructure and hardware (Configuration with A100: single GPU) on the German NER split of XTREAM, enabling a direct comparison.

**Quality vs. energy efficiency.** Despite requiring no task-specific fine-tuning, Gemma-3-4b-it outperforms GoLLIE on German NER by a substantial margin (F1 = 0.616 vs. 0.521, +18%) while consuming *less* GPU energy (37,807 J vs. 46,654 J, $-19\%$). The combined effect is a 50% advantage in quality-per-joule (F1/J: $1.67 \times 10^{-5}$ vs. $1.12 \times 10^{-5}$). GoLLIE does achieve notably higher throughput (59.8 Tok/s vs. 14.2 Tok/s for Gemma-3-4b-it), because its code-style output format is far more compact than the natural-language explanations produced by instruction-tuned models. However, this latency advantage does not translate into an energy advantage under our batch-serving regime, where total joules are dominated by the prefill cost of the input context rather than the generation length.

**Scaling penalty for large generalist models.** Scaling up the instruction-tuned model beyond 4B parameters erodes the energy advantage rapidly. Gemma-3-12b-it and Gemma-3-27b-it reach higher German F1 scores (0.712 and 0.706, respectively) but at 65% and 3.3× more energy than GoLLIE,

| Task | Hardware | Model | $C$ | F1 | P | R | E (kJ) | J/TP |
|---|---|---|---|---|---|---|---|---|
| XTREME | Alpha | 4B | 256 | 0.531 | 0.557 | **0.513** | 606.7 | 21.0 |
| | | | 512 | 0.492 | 0.522 | 0.474 | 445.1 | 15.2 |
| | | | 1024 | 0.508 | 0.532 | 0.494 | 402.0 | 13.3 |
| | | 12B | 256 | 0.566 | 0.592 | **0.550** | 1397.9 | 40.4 |
| | | | 512 | 0.550 | 0.584 | 0.533 | 1056.4 | 40.4 |
| | | | 1024 | 0.545 | 0.586 | 0.527 | 1037.3 | 42.1 |
| | | 27B | 256 | 0.574 | 0.601 | 0.557 | 1591.9 | 54.9 |
| | | | 512 | 0.551 | 0.567 | 0.544 | 1890.5 | 67.2 |
| | | | 1024 | 0.561 | 0.571 | **0.559** | 1728.3 | 3107.4 |
| | | 70B | 256 | 0.357 | 0.494 | **0.286** | 4116.7 | 237.6 |
| | | | 512 | 0.352 | 0.492 | 0.281 | 2948.0 | 206.4 |
| | | | 1024 | 0.347 | 0.490 | 0.276 | 2678.0 | 188.1 |
| | Capella | 4B | 256 | 0.463 | 0.500 | **0.443** | 26.9 | 10.9 |
| | | | 512 | 0.425 | 0.501 | 0.385 | 31.0 | 15.8 |
| | | | 1024 | 0.401 | 0.501 | 0.349 | 28.3 | 15.0 |
| | | 12B | 256 | 0.490 | 0.558 | **0.471** | 55.4 | 41.5 |
| | | | 512 | 0.458 | 0.559 | 0.414 | 47.9 | 19.0 |
| | | | 1024 | 0.456 | 0.558 | 0.404 | 64.3 | 36.0 |
| | | 27B | 256 | 0.487 | 0.546 | **0.471** | 94.7 | 37.1 |
| | | | 512 | 0.442 | 0.547 | 0.405 | 86.4 | 45.7 |
| | | | 1024 | 0.483 | 0.547 | 0.454 | 101.5 | 41.9 |
| | | 70B | 256 | 0.478 | 0.589 | 0.440 | 59.2 | 40.2 |
| | | | 512 | 0.569 | 0.679 | **0.509** | 55.8 | 10.6 |
| | | | 1024 | 0.561 | 0.678 | 0.498 | 57.6 | 12.1 |
| Masakha | Alpha | 4B | 256 | 0.552 | 0.553 | **0.557** | 115.1 | 6.7 |
| | | | 512 | 0.536 | 0.542 | 0.537 | 73.1 | 4.9 |
| | | | 1024 | 0.525 | 0.526 | 0.534 | 61.6 | 5.4 |
| | | 12B | 256 | 0.653 | 0.637 | **0.678** | 409.2 | 27.5 |
| | | | 512 | 0.633 | 0.622 | 0.651 | 299.3 | 15.0 |
| | | | 1024 | 0.638 | 0.620 | 0.665 | 215.0 | 12.0 |
| | | 27B | 256 | 0.708 | 0.690 | **0.732** | 540.9 | 29.1 |
| | | | 512 | 0.700 | 0.693 | 0.716 | 337.0 | 18.2 |
| | | | 1024 | 0.694 | 0.684 | 0.711 | 417.1 | 22.4 |
| | | 70B | 256 | 0.705 | 0.735 | **0.685** | 1702.0 | 96.6 |
| | | | 512 | 0.697 | 0.734 | 0.671 | 1471.7 | 86.3 |
| | | | 1024 | 0.690 | 0.724 | 0.666 | 1561.6 | 90.1 |
| EE | Alpha | 4B | 256 | 0.290 | 0.289 | 0.290 | 57.6 | 11.7 |
| | | | 512 | 0.285 | 0.285 | 0.285 | 84.2 | 71.4 |
| | | | 1024 | 0.274 | 0.255 | **0.296** | 71.5 | 75.4 |
| | | 12B | 256 | 0.353 | 0.359 | **0.347** | 29.1 | 14.9 |
| | | | 512 | 0.343 | 0.357 | 0.330 | 25.7 | 13.8 |
| | | | 1024 | 0.340 | 0.361 | 0.321 | 16.8 | 12.2 |
| | | 27B | 256 | 0.346 | 0.347 | **0.344** | 54.0 | 30.6 |
| | | | 512 | 0.339 | 0.347 | 0.332 | 72.2 | 40.8 |
| | | | 1024 | 0.336 | 0.345 | 0.328 | 82.7 | 32.0 |
| | | 70B | 256 | 0.345 | 0.364 | 0.329 | 201.4 | 33.5 |
| | | | 512 | 0.340 | 0.329 | **0.352** | 266.0 | 43.6 |
| | | | 1024 | 0.341 | 0.359 | 0.324 | 168.4 | 28.9 |
| | Capella | 4B | 256 | 0.208 | 0.258 | 0.193 | 5.22 | 15.3 |
| | | | 512 | 0.267 | 0.259 | 0.278 | 4.37 | 7.2 |
| | | | 1024 | 0.268 | 0.259 | **0.280** | 4.33 | 7.1 |
| | | 12B | 256 | 0.213 | 0.312 | 0.165 | 5.16 | 17.1 |
| | | | 512 | 0.228 | 0.312 | 0.186 | 7.74 | 21.8 |
| | | | 1024 | 0.325 | 0.310 | **0.346** | 13.64 | 18.0 |
| | | 27B | 256 | 0.306 | 0.331 | 0.294 | 19.34 | 29.1 |
| | | | 512 | 0.326 | 0.335 | 0.320 | 25.57 | 36.9 |
| | | | 1024 | 0.325 | 0.331 | **0.322** | 26.76 | 38.4 |
| | | 70B | 256 | 0.328 | 0.324 | **0.336** | 24.25 | 32.9 |
| | | | 512 | 0.326 | 0.323 | 0.331 | 24.14 | 33.2 |
| | | | 1024 | 0.327 | 0.324 | 0.332 | 25.30 | 34.7 |
| RE | Alpha | 4B | 256 | 0.102 | 0.178 | 0.071 | 25.5 | 29.5 |
| | | | 512 | 0.095 | 0.107 | 0.085 | 53.2 | 51.5 |
| | | | 1024 | 0.096 | 0.108 | **0.086** | 53.1 | 50.7 |
| | | 12B | 256 | 0.167 | 0.178 | **0.157** | 313.9 | 164.2 |
| | | | 512 | 0.157 | 0.174 | 0.144 | 97.4 | 55.8 |
| | | | 1024 | 0.168 | 0.179 | **0.157** | 281.0 | 146.9 |
| | | 27B | 256 | 0.223 | 0.239 | **0.209** | 295.0 | 116.1 |
| | | | 512 | 0.218 | 0.238 | 0.201 | 475.3 | 194.1 |
| | | | 1024 | 0.161 | 0.207 | 0.132 | 224.4 | 140.1 |
| | | 70B | 256 | 0.236 | 0.259 | **0.216** | 899.7 | 342.6 |
| | | | 512 | 0.233 | 0.258 | 0.212 | 578.8 | 224.8 |
| | | | 1024 | 0.234 | 0.259 | 0.213 | 647.4 | 250.1 |
| | Capella | 4B | 256 | 0.067 | 0.080 | **0.058** | 24.04 | 37.5 |
| | | | 512 | 0.060 | 0.076 | 0.050 | 22.29 | 39.5 |
| | | | 1024 | 0.065 | 0.084 | 0.054 | 21.17 | 39.2 |
| | | 12B | 256 | 0.147 | 0.170 | **0.130** | 49.85 | 32.3 |
| | | | 512 | 0.147 | 0.170 | **0.130** | 48.49 | 31.5 |
| | | | 1024 | 0.147 | 0.171 | **0.130** | 48.15 | 31.1 |
| | | 27B | 256 | 0.167 | 0.205 | **0.137** | 22.2 | 13.3 |
| | | | 512 | 0.146 | 0.199 | 0.116 | 20.91 | 16.1 |
| | | | 1024 | 0.086 | 0.144 | 0.062 | 21.50 | 28.7 |
| | | 70B | 256 | 0.185 | 0.204 | 0.170 | 73.88 | 37.3 |
| | | | 512 | 0.187 | 0.206 | **0.172** | 74.65 | 37.2 |
| | | | 1024 | 0.187 | 0.206 | **0.172** | 75.27 | 37.5 |

Table 8: Effect of concurrency $C$ across tasks and models, evaluated on Alpha and Capella partitions.

pushing their F1/J ($6.6 \times 10^{-6}$) well below GoLLIE's. LLaMA-3.3-70B presents the starkest trade-off: it uses $7.3\times$ more energy than GoLLIE yet achieves *lower* F1 (0.471 vs. 0.521), making it the least energy-efficient option by a wide margin. These results suggest that for NER on A100 hardware, a compact 4B instruction-tuned model occupies a Pareto-optimal position that neither fine-

tuned specialists nor Large-scale generalists can match without task-specific adaptation.

**Implications for deployment.** The comparison reinforces a practically important finding: the energy cost of fine-tuning a specialist model such as GoLLIE (which is a one-time, offline expense) can be amortised across many inference calls, but the *per-query* Energy at inference time is what dominates in production. Our results show that, at inference time, a zero-shot 4B model is already more energy-efficient than a 7B fine-tuned specialist on the same NER task. If the serving configuration is further optimised using the recommendations in specifically `max_num_seqs=512` and `max_new_tokens=100` for NER - the gap widens: Gemma-3-4b-it would consume an estimated additional 12–20% less energy, pushing its advantage over GoLLIE to roughly $2\times$ in F1/J.

### A.10 Hyperparameter Optimisation vs Single-Parameter Sweeps

| Parameter | Value | Mean $J_{\text{TP}}^{\text{corr}}$ | Min $J_{\text{TP}}^{\text{corr}}$ | $\Delta$ vs BL (%) |
|---|---|---|---|---|
| kv_cache_dtype | auto | 4.41 | 4.34 | -15.5 |
| | fp8 (BL) | 5.22 | 4.54 | 0.0 |
| block_size | 16 (BL) | 5.22 | 4.54 | 0.0 |
| | 32 | 34.78 | 5.60 | +566.2 |
| enable_prefix_caching | True (BL) | 5.22 | 4.54 | 0.0 |
| | False | 4.62 | 4.52 | -11.5 |
| chunked_prefill | True (BL) | 5.22 | 4.54 | 0.0 |
| | False | 5.32 | 4.88 | +1.9 |
| enforce_eager | False (BL) | 5.22 | 4.54 | 0.0 |
| | True | 6.67 | 6.55 | +27.7 |
| max_new_tokens | 50 | 5.83 | 5.64 | +11.7 |
| | 100 (BL) | 5.22 | 4.54 | 0.0 |
| | 150 | 28.45 | 5.17 | +445.0 |
| | 200 | 27.87 | 4.36 | +433.8 |

Table 9: One-Parameter-at-a-Time (OPAAT) sweep results. Baseline (BL) is indicated. $\Delta$ is the relative change of the mean $J_{\text{TP}}^{\text{corr}}$ against the baseline.

| Hyperparameter | Optimal Value |
|---|---|
| block_size | 32 |
| chunked_prefill_enabled | False |
| enable_prefix_caching | False |
| enforce_eager | False |
| kv_cache_dtype | auto |
| max_new_tokens | 150 |
| **Best Objective ($J_{\text{TP}}^{\text{corr}}$)** | **4.25 J/TP (-18.6% vs BL)** |

Table 10: The globally optimal configuration discovered by the Optuna HPO engine after 56 trials.

To minimise the corrected Energy per True Positive ($J/TP$) during inference, we evaluated the serving configuration of the 4B model using two distinct search strategies: a One-Parameter-at-a-Time (OPAAT) sweep and Bayesian Hyperparameter Optimisation (HPO) via Optuna. The OPAAT approach isolates individual variables against a default baseline, successfully identifying immediate efficiency gains. For instance, switching the KV cache data type to `auto` and disabling prefix caching yielded energy reductions of 15.5% and 11.5%, respectively. However, this isolated approach obscures critical inter-parameter dynamics and can lead to flawed conclusions about the utility of absolute parameters.

| Hyperparameter | Importance Variance (%) |
|---|---|
| max_new_tokens | 43.1 |
| enforce_eager | 21.3 |
| chunked_prefill_enabled | 18.0 |
| enable_prefix_caching | 10.6 |
| block_size | 5.5 |
| kv_cache_dtype | 1.5 |

Table 11: Global parameter importance calculated via fANOVA over the HPO trial history. Values indicate the percentage of variance in $J_{\text{TP}}^{\text{corr}}$ explained by each parameter.

The Bayesian HPO strategy overcomes this limitation by exploring the joint configuration space, ultimately uncovering a global optimum that reduces the energy metric by 18.6% relative to the baseline. Crucially, HPO exposes hidden, non-linear interactions between variables. A prime example is the `block_size` parameter: in the isolated OPAAT sweep, setting `block_size=32` resulted in a catastrophic 566% energy penalty. Yet the HPO engine determined that, when combined with disabled chunked prefill and an `auto` KV cache, `block_size=32` is a necessary component of the most energy-efficient configuration.

Furthermore, variance analysis (fANOVA) of the HPO trials completely reordered the assumed hierarchy of parameter importance. While the OPAAT sweep suggested that the KV cache data type was the critical lever for energy savings, fANOVA revealed it accounts for the lowest global importance (1.5%) across the joint space. Instead, maximum token limits and the enforcement of eager execution drove the vast majority of performance variance. Ultimately, the HPO search yielded a marginal 3.7% efficiency gain over the best OPAAT combination. Because this nominal improvement falls well within the natural run-to-run variance of GPU telemetry and workload scheduling, the substantial computational overhead required for joint optimisation is essentially unjustified.

### A.11 Optimal Serving Configurations

Table 12 distils the one-parameter-at-a-time ablation results into eight actionable serving-layer settings; each row states the recommended value, its target task scope, and the median energy and quality-per-energy shifts relative to the shared base-

line configuration.

### A.12 Influence of the Formats and Decoding

Table 13 reports, for each task and model, the baseline F1 and energy under unconstrained JSON decoding, along with the best-performing output format, quantifying the F1 gain and energy change achieved by switching to the optimal format.

### A.13 Comparison of the Architectures and Distribution Methods

**Baseline energy efficiency across hardware architectures.** Under a shared baseline (`kv_cache_dtype=fp8`, `block_size=16`, `max_new_tokens=150`, `batch_size=1024`), the H100 consumes substantially less energy than the A100. For 12B multilingual NER (XTREME), the H100 expends 77,851 J versus 196,416 J on the A100—a 2.5-fold reduction for an equivalent 0.507 $F_1$ score, yielding a proportionally higher $F_1/\text{kJ}$. This advantage peaks at smaller scales: the 4B model achieves 0.039 and 0.071 $F_1/\text{kJ}$ for Event Extraction (EE) and African NER (MASAKHA) on H100, degrading to 0.007 and 0.032 $F_1/\text{kJ}$ at 27B. Conversely, the A100 12B matches H100 12B efficiencies (0.032 $F_1/\text{kJ}$ for EE/MASAKHA) despite higher total energy, indicating the architectural gap narrows when workloads saturate memory bandwidth. Across tasks, the 4B H100 consistently delivers the highest $F_1/\text{kJ}$, confirming its memory bandwidth and tensor-core density disproportionately benefit throughput-bound inference.

**Divergent parameter sensitivity between GPU generations.** H100 and A100 exhibit divergent responses to serving-layer tuning. KV cache quantization (FP8 vs. BF16) shows the starkest contrast: on H100, FP8 uniformly cuts energy by 10–16% across EE, Relation Extraction (RE), and XTREME, boosting $F_1/\text{kJ}$ by $+11\%$ to $+21\%$. On A100, FP8 similarly benefits EE and RE but harms MASAKHA ($+11\%$ energy, $-14.2\%$ $F_1/\text{kJ}$), likely due to the precision-sensitive token distributions of morphologically complex languages. Block size behavior also diverges: H100 heavily penalizes sizes above 16 (`block_size=128` drops $F_1/\text{kJ}$ by $-30.7\%$), while A100 benefits from sizes 32 and 64 ($+37.9\%$ and $+23.5\%$ $F_1/\text{kJ}$ for EE), reflecting generational shifts in paged-attention overhead. Notably, `batch_size=256` is the only universally optimal setting, slashing energy ($-22.9\%$ H100, $-16.2\%$ A100) and surging $F_1/\text{kJ}$ ($+31.8\%$ H100, $+50.4\%$ A100), making it the most impactful architecture-agnostic tuning lever.

**GPU inference cost ($J/\text{TP}$) analysis and cross-task implications.** The Joules per True Positive ($J/\text{TP}$) metric quantifies extraction cost independently of recall-precision trade-offs. On H100, NER ranges from 4.45 J/TP (4B, DST) to 9.09 J/TP (27B, DST). Crucially, the 70B model remains highly competitive at 4.94 J/TP, proving H100 throughput allows massive architectures to recover true positives at costs comparable to 4B models—unlike the A100, where the cross-scale energy gap is much wider. For EE, $J/\text{TP}$ varies from 7.22 (4B, DST) to 23.74 (27B, DST), a 3.3-fold spread driven by generation length rather than extraction quality. RE exhibits the widest disparity: structured JSON at 12B requires 25.1 J/TP, while unstructured DST inflates to 353 J/TP—a 14-fold penalty caused by near-zero true-positive yields in free-form generation. Ultimately, the H100's efficiency advantage concentrates in two regimes: deploying small models with concise formats for high-volume NER, and using grammar-constrained decoding for structured extraction, where H100 compute density absorbs the constrained-generation overhead with a significantly lower energy tax than A100.

### A.14 GPU Utilisation and Energy Efficiency

In throughput-bound LLM serving, GPU utilisation is proxied by the number of concurrent requests ($C$). Low $C$ starves the decoding engine, burning energy without generating tokens. As $C$ increases, the scheduler keeps tensor cores occupied, amortising per-request overhead. We exploit a concurrency sweep ($C \in \{256, 512, 1024\}$) across hardware configurations to evaluate this relationship via $F_1/\text{kJ}$ and J/TP.

Table 14 reports efficiency across concurrency levels. Three patterns emerge. First, scaling $C$ from 256 to 1024 universally improves J/TP and $F_1/\text{kJ}$, with the largest gains at the 4B scale. For A100 (XTREME), the 4B model's J/TP drops from 21.0 to 13.3 ($-37\%$), while $F_1/\text{kJ}$ rises 44%. On H100 (MASAKHA), the same model improves from 7.6 to 5.0 J/TP ($-34\%$); this smaller gain reflects the H100's higher base throughput. Second, energy savings are super-linear at low $C$: doubling from 256 to 512 cuts A100 MASAKHA 12B energy by 27% (409.2→299.3 kJ), and doubling to

| Parameter Configuration | Thermodynamic & Predictive Impact |
|---|---|
| **`max_num_seqs` = 512** *(All Tasks)* | Reduces absolute Energy(J) by 12% and increases overall efficiency by $+15\%$ $F_1$/J, with negligible ($< 1\%$) degradation in absolute $F_1$. |
| **`batch_size` = 256** *(EE / RE)* | Alleviates memory bandwidth contention. Saves $\sim$22% in Energy(J) and improves overall $F_1$/J by $\sim$32% by optimizing scheduler fill-rates. |
| **`max_new_tokens` = 100** *(NER Tasks)* | Eliminates post-extraction over-generation. Lowers $J/\text{Gen}$ and reduces absolute Energy(J) by 12%, while yielding a $+20\%$ surge in $F_1$/J. |
| **`max_new_tokens` = 200** *(Relation Extraction)* | Optimally bounds complex reasoning to minimize $J/\text{TP}$. Saves 24% in Energy(J) and delivers a massive $+70\%$ improvement in $F_1$/J. |
| **`max_model_len` = 2048** *(Relation Extraction)* | Matches doc-level context. Recovers $F_1$ lost to truncation while maintaining an efficient energy balance ($+13\%$ $F_1$/J). |
| **`max_num_batched_tokens` = 32768** *(Multilingual NER)* | Packs short prompts more efficiently. Improves $F_1$/J by $\sim$4% by reducing prefill overhead without impacting accuracy. |
| **`block_size` = 64** *(All Tasks)* | Modestly reduces memory fragmentation. Decreases Energy(J) by $\sim$5% and yields a low-risk $+8\%$ improvement in $F_1$/J without altering accuracy. |
| **`kv_cache_dtype` = auto** *(A100 Deployments)* | Prevents the 20–40% throughput penalty observed with `fp8`. Defaulting to `fp8` artificially inflates $J/\text{entity}$ and causes a penalty in $F_1$/J. |

Table 12: Summary of optimal serving configurations and their median thermodynamic/predictive shifts against the baseline.

| Task | Model | BL F1 | Best F1 | ΔF1 (%) | Best Format | BL E (J) | ΔE (%) |
|---|---|---|---|---|---|---|---|
| NER | Gemma-3-4B | 0.4492 | **0.4898** | **+9.0** | $\text{DST}_{\varnothing}$ | 31,383 | −4.7 |
| | Gemma-3-12B | 0.4924 | **0.5175** | **+5.1** | $\text{YAML}_{\varnothing}$ | 60,871 | −2.1 |
| | Gemma-3-27B | 0.4791 | **0.5028** | **+4.9** | $\text{JSON}_{\text{xgr}}$ | 85,685 | −3.9 |
| | Llama-3.3-70B | 0.4777 | **0.5082** | **+6.4** | $\text{JSON}_{\text{xgr}}$ | 59,210 | −10.0 |
| EE | Gemma-3-4B | 0.2449 | **0.2630** | **+7.4** | $\text{JSON}_{\text{xgr}}$ | 674 | +12.9 |
| | Gemma-3-12B | 0.1861 | **0.3238** | **+74.0** | $\text{JSON}_{\text{xgr}}$ | 645 | +144.1 |
| | Gemma-3-27B | 0.3277 | 0.3277 | +0.0 | $\text{JSON}_{\varnothing}$ | 2,850 | +0.0 |
| | Llama-3.3-70B | 0.3282 | 0.3282 | +0.0 | $\text{JSON}_{\varnothing}$ | 3,025 | +0.0 |
| RE | Gemma-3-4B | 0.0666 | 0.0666 | +0.0 | $\text{JSON}_{\varnothing}$ | 3,005 | +0.0 |
| | Gemma-3-12B | 0.1472 | **0.1479** | **+0.5** | $\text{JSON}_{\text{xgr}}$ | 5,382 | +31.0 |
| | Gemma-3-27B | 0.1364 | **0.1783** | **+30.7** | $\text{JSON}_{\text{xgr}}$ | 26,370 | −52.2 |
| | Llama-3.3-70B | 0.1855 | 0.1855 | +0.0 | $\text{JSON}_{\varnothing}$ | 9,214 | +0.0 |

Table 13: Cross-task summary: best format and decoding regime per task and model. Baseline (BL) utilises unconstrained JSON. Best format denotes the highest mean F1 across all evaluated modes. The unit of energy (E) is the Joule. Bold cells mark cases where an alternative format improves over baseline.

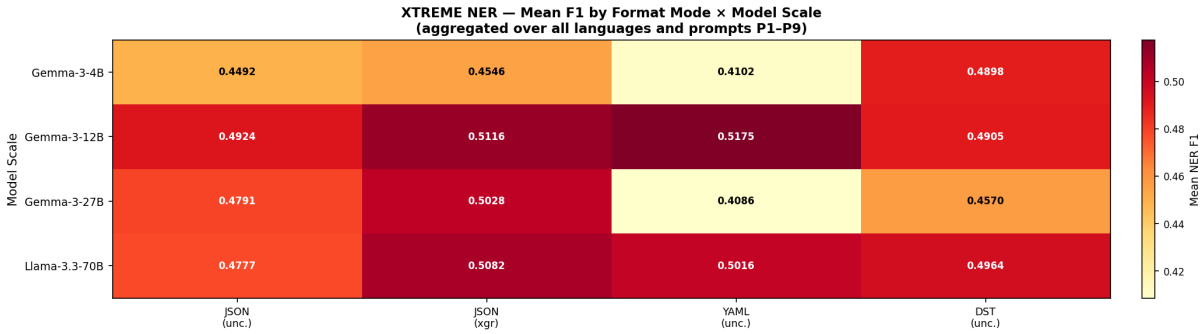


Figure 5: Influence of the format and decoding regimes on the energy consumption across models for the XTREAM NER task

1024 saves another 28%, confirming GPU under-utilisation at $C$=512. Third, scaling incurs a mild quality trade-off: the 70B model loses only 0.01 $F_1$ points, while the 4B model loses up to 0.04 $F_1$ points, reflecting the larger model's capacity to handle KV cache pressure.

The data confirms a monotone positive correlation between concurrency and efficiency. However, gains saturate differently by scale: 70B J/TP improves only 9% from $C$=512 to 1024, versus 37% for 4B, as the larger KV cache footprint occupies High Bandwidth Memory (HBM) more consis-

| HW | Task | Mod. | $C$ | $F_1$ | $E$ (kJ) | $F_1/\text{kJ} \uparrow$ | J/TP$\downarrow$ |
|---|---|---|---|---|---|---|---|
| A100 | XTREME | 4B | 256 | 0.531 | 606.7 | $8.76\times10^{-4}$ | 21.0 |
| | | | 512 | 0.492 | 445.1 | $1.11\times10^{-3}$ | 15.2 |
| | | | 1024 | 0.508 | 402.0 | $1.26\times10^{-3}$ | 13.3 |
| | | 70B | 256 | 0.357 | 4116.7 | $8.67\times10^{-5}$ | 237.6 |
| | | | 512 | 0.352 | 2948.0 | $1.19\times10^{-4}$ | 206.4 |
| | | | 1024 | 0.347 | 2678.0 | $1.30\times10^{-4}$ | 188.1 |
| | MASAKHA | 4B | 256 | 0.552 | 115.1 | $4.80\times10^{-3}$ | 6.7 |
| | | | 512 | 0.536 | 73.1 | $7.34\times10^{-3}$ | 4.9 |
| | | | 1024 | 0.525 | 61.6 | $8.52\times10^{-3}$ | 5.4 |
| | | 12B | 256 | 0.653 | 409.2 | $1.60\times10^{-3}$ | 27.5 |
| | | | 512 | 0.633 | 299.3 | $2.11\times10^{-3}$ | 15.0 |
| | | | 1024 | 0.638 | 215.0 | $2.97\times10^{-3}$ | 12.0 |
| H100 | MASAKHA | 4B | 256 | 0.441 | 54.9 | $8.04\times10^{-3}$ | 7.6 |
| | | | 512 | 0.464 | 50.6 | $9.17\times10^{-3}$ | 5.1 |
| | | | 1024 | 0.468 | 48.0 | $9.74\times10^{-3}$ | 5.0 |

Table 14: Efficiency metrics across concurrency ($C$), tasks, and models. $E$ is the total corrected energy across all languages. $\uparrow$ / $\downarrow$ indicates the desired direction.

tently. The H100 achieves higher absolute $F_1/\text{kJ}$ and lower energy across all levels, reflecting superior bandwidth and compute density. Practitioners should maximise $C$ to the hardware memory ceiling (e.g., 1024 for 4B–12B; 512 for 70B), accepting minor $F_1$ penalties ($\leq$0.04) for disproportionate energy savings (25–47%) and J/TP reductions (34–37%).