machine learning & knowledge extraction

MDPI

Article

Evaluating the Impact of Adapter-Based Fine-Tuning on Structured Parsing Performance in Large Language Models

Ratomir Karlović [1], Luka Sever [1], Sandi Baressi Šegota [1], Vedran Mrzljak [2,*] and Ivan Lorencin [1]

[1] Faculty of Informatics, Juraj Dobrila University of Pula, Zagrebačka 30, 52100 Pula, Croatia; ratomir.karlovic@unipu.hr (R.K.); luka.sever@unipu.hr (L.S.); sandi.baressi.segota@unipu.hr (S.B.Š.); ivan.lorencin@unipu.hr (I.L.)
[2] Faculty of Engineering Rijeka, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia
* Correspondence: vedran.mrzljak@riteh.uniri.hr

Abstract

Recent advances in large language models (LLMs) highlight two dominant strategies for performance improvement: prompt engineering and fine-tuning. While prompt design can significantly influence model output, it remains uncertain whether lightweight fine-tuning methods, such as adapter-based training, offer meaningful advantages for structured, domain-specific tasks. This study builds on prior research comparing three prompting strategies for natural-language command parsing into JSON schemas. Expanding that framework, the current work investigates how adapter-based fine-tuning, where most model parameters are frozen and only small adapter modules are trained, affects model accuracy and consistency. The experiment uses the same controlled shopping-cart parsing task and dataset of 12,000 synthetic commands to ensure direct comparability. Results quantify the trade-off between computational cost and performance gains, offering evidence-based insights into whether fine-tuning is a justified investment compared to advanced prompt engineering. The contribution of this study is a clear, empirical framework for deciding when fine-tuning meaningfully enhances LLM utility in applied natural-language understanding.

Keywords: LLM; adapters fine-tuning; prompt engineering; natural language understanding; JSON parsing; structured output; model performance

1. Introduction

The rapid evolution of Large Language Models (LLMs) has fundamentally transformed the landscape of Natural Language Processing (NLP), enabling advanced capabilities in reasoning, content generation, and semantic understanding [1]. While early applications focused primarily on open-ended text generation, modern production environments increasingly require LLMs to function as reliable semantic parsers capable of mapping unstructured user intent into rigorous, machine-readable formats such as JSON or SQL [2]. This capability is particularly critical in domain-specific applications, such as e-commerce controllers, where valid schema adherence is a prerequisite for system functionality.

A primary challenge in deploying these systems is the "optimization dilemma": choosing between inference-time strategies and model training. As highlighted in our previous survey [3], while prompt engineering offers a lightweight "black-box" approach to adaptation, it often struggles with robustness when constrained to rigid schema definitions. Conversely, full-parameter fine-tuning yields high precision but incurs prohibitive computational costs and memory requirements [4]. To address this, Parameter-Efficient Fine-Tuning

Check for updates

Academic Editor: Irena Spasić

Received: 13 April 2026
Revised: 15 May 2026
Accepted: 18 May 2026
Published: 21 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

(PEFT) techniques, such as Low-Rank Adaptation (LoRA), have emerged as a middle ground, freezing pre-trained weights and injecting trainable rank-decomposition matrices to approximate full fine-tuning performance with a fraction of the resources [5].

To address these challenges, this study evaluates the cost-benefit ratio of fine-tuning against prompting for high-fidelity semantic parsing. Building upon the theoretical frameworks established in our prior work [3], which primarily provided a broad qualitative overview of LLM capabilities and adaptation methodologies, the specific novelty of this current research lies in conducting a targeted, empirical comparison. We move beyond theoretical discussions to directly quantify the performance and computational trade-offs of PEFT versus prompt engineering in a highly constrained generative environment. We implement a controlled experimental design using multiple models. The system compares two distinct optimization paradigms: Advanced Prompt Engineering and Adapter-Based Fine-Tuning, utilizing LoRA to update the model's internal representations [6]. This setup allows for a direct comparison of how each method handles the rigorous constraints of mapping natural language commands to structured JSON outputs.

The main aim of this research is to examine the trade-offs between prompt engineering and PEFT when applied to structured parsing tasks. In particular, the study investigates how different adaptation strategies influence the ability of LLMs to generate accurate and valid structured outputs. The primary contribution of this work is a structured empirical evaluation framework for benchmarking multiple models and adaptation approaches on structured parsing tasks. It is crucial to emphasize that this framework is evaluated using a controlled, synthetic benchmark featuring a relatively simple, three-attribute JSON schema. While this design isolates the fundamental mechanisms of structured extraction from the unpredictable noise of real-world inputs—such as typos, multi-intent requests, and complex nested structures—it inherently limits immediate generalization to highly unstructured or out-of-distribution data. Consequently, the findings presented herein should be interpreted as foundational, comparative baselines rather than definitive performance guarantees for all real-world deployments. Ultimately, the results provide practical insights into the relative effectiveness of prompting strategies and adapter-based methods, highlighting scenarios in which lightweight fine-tuning can offer measurable improvements in reliability compared to prompt-only configurations.

2. Background and Related Works

The rapid growth of LLMs has necessitated strategies for adapting them efficiently to downstream tasks. Zhiqiang Hu et al. [5] introduced LLM-Adapters, a framework for PEFT that integrates Series adapters, Parallel adapters, Prefix-Tuning, and LoRA. Their study demonstrated that smaller LLaMA models (7B and 13B) equipped with optimized adapters could match or surpass the performance of much larger models such as ChatGPT 5.2 and GPT-3.5, particularly on structured reasoning tasks using datasets like Math10K and Commonsense170K. These findings suggest that adapter-based PEFT can improve structured task accuracy compared to advanced prompting strategies.

Ding et al. [7] proposed a unified framework called delta-tuning, which adjusts only a small fraction of model parameters. Through extensive evaluation on over 100 NLP tasks, they found that delta-tuning achieved performance comparable to full fine-tuning while significantly reducing GPU memory usage and accelerating backpropagation. Larger models not only benefited more from minimal parameter updates but also converged faster, reinforcing the potential of adapter-based methods to enhance task-specific performance efficiently.

Esmaeili et al. [8] conducted empirical studies on PEFT for code-specific LLMs, examining LoRA, Compacter, and Infused Adapter by Inhibiting and Amplifying Inner Activations

(IA^3). Their findings showed that LoRA consistently outperformed other adapters in code summarization and generation, occasionally exceeding full fine-tuning on low-resource languages like R. The study highlighted that the number of trainable parameters influenced functional correctness more than the adapter architecture itself, further supporting the effectiveness of PEFT in improving structured outputs in specialized domains.

Razuvayevskaya et al. [9] compared parameter-efficient techniques and full fine-tuning for multilingual news classification. Using XLM-RoBERTa Large, they showed that LoRA and bottleneck adapters could reduce trainable parameters by 140–280 times while maintaining competitive accuracy. Their results suggested that PEFT strategies can outperform traditional prompting approaches, especially when sufficient source data is available.

Shin et al. [10] assessed the trade-offs between prompt engineering and fine-tuning for automated code tasks. Their evaluation of GPT-4 against seventeen fine-tuned baselines revealed that while task-specific prompting sometimes outperformed automated methods in code summarization, fine-tuned models were generally superior in code generation. Human-in-the-loop conversational prompting further enhanced outcomes, demonstrating that adapter-based fine-tuning can stabilize and improve output quality over prompting alone.

Ming Gong et al. [11] developed structure-learnable adapters, which introduced differentiable gating and sparsity control to dynamically optimize adapter placement and activation paths. Experiments on the Multi-Task NLU Benchmark indicated that this approach not only achieved high accuracy with minimal parameters but also reduced output variability and maintained robustness under noisy conditions, supporting the notion that adapter fine-tuning can yield more consistent outputs than prompting.

Fang and Ye [12] proposed RFedLR, a robust PEFT framework for federated learning. By selectively updating noise-sensitive parameters and dynamically weighting client updates, RFedLR achieved up to 10.15% accuracy improvements over state-of-the-art baselines under high-noise conditions while using only 0.1754% of the trainable parameters. These results reinforce that adapter-based fine-tuning provides more stable and reliable outputs than prompt-based methods, even in decentralized and noisy environments.

Shuo Chen et al. [13] benchmarked eleven adaptation methods on vision-language models under textual and visual corruptions. They found that parameter-efficient adapters exhibited higher resilience to input perturbations than full fine-tuning or prompting, although no single method dominated across all datasets. Increasing adaptation data or parameter size did not guarantee robustness, highlighting the importance of careful PEFT design to enhance model stability.

Chen et al. [14] evaluated prompt engineering for industrial document processing tasks and found that few-shot learning improved reasoning capabilities over zero-shot prompts. However, adapter-based PEFT consistently offered more reliable and less variable outputs, particularly in complex scenarios with noisy OCR inputs.

Kaijie Zhu et al. [15] introduced PromptRobust, a benchmark for adversarial prompt evaluation across multiple LLMs. Their results showed that word-level adversarial prompts caused a 39% average performance drop, and few-shot prompts exhibited better robustness than zero-shot prompting. The study illustrates that adapter fine-tuning can improve resilience to prompt-based perturbations.

Jaehyung Kim et al. [16] presented ROAST, combining adversarial perturbations with selective training to improve robustness across sentiment classification and entailment tasks. ROAST yielded average improvements of 18.39% and 7.63% over state-of-the-art baselines, demonstrating that adapter fine-tuning can mitigate input noise effects more effectively than prompting.

Pei et al. [17] proposed SelfPrompt, which autonomously evaluates LLM robustness using domain-constrained knowledge graphs and adversarial prompt generation. Their findings indicated that domain-specific adapter fine-tuning enhances robustness in specialized fields like medicine and biology, outperforming prompting strategies alone.

Lin Mu et al. [18] introduced Robustness of Prompting (RoP), a parameter-free method for improving LLM resilience through error correction and guidance prompts. While effective, experiments confirmed that adapter fine-tuning provides superior stability and transferability across architectures, emphasizing the practical value of PEFT in noisy real-world environments.

Sajjadi Mohammadabadi et al. [19] surveyed LLM architectures and adaptation methods, emphasizing the efficiency of PEFT, instruction tuning, and RLHF. They noted that adapter-based fine-tuning can deliver significant performance gains relative to prompting, justifying its computational cost in applied settings.

Trad and Chehab [20] studied phishing detection LLMs and found that while refined prompting improved performance, fine-tuned models achieved higher F1 Scores and superior reliability, demonstrating that the cost of PEFT is justified when high precision is required.

Pornprasit and Tantithamthavorn [21] evaluated LLM-based code review, showing that fine-tuning GPT-3.5 achieved 73–74% higher Exact Match rates than prompting approaches. Their study confirmed that adapter-based fine-tuning can deliver meaningful gains even with modest training data.

Wang et al. [22] introduced COSMOS, a framework for predicting adaptation outcomes with minimal trials. Their results indicated that fine-tuning consistently outperformed in-context learning (ICL) in medium to high-cost scenarios, supporting the view that adapter PEFT justifies its resource investment by providing predictable and superior performance.

Structured Output Generation and Constrained Decoding

To facilitate reliable integration into developer workflows, Liu et al. identified a critical need for constraining LLM outputs, categorizing requirements into low-level structural formats like valid JSON and high-level semantic guidelines [23]. Evaluating model proficiency in generating these structured outputs has led to dedicated benchmarks. Geng et al. introduced JSONSchemaBench to systematically evaluate frameworks handling complex schema constraints [24]. Similarly, Shorten et al. developed StructuredRAG, revealing that zero-shot prompting strategies often fail as task complexity increases, necessitating advanced prompt optimization to achieve reliable formatting [25]. To actively improve these capabilities, Lu et al. proposed SchemaBench alongside a Schema Reinforcement Learning framework, utilizing a fine-grained validator to provide dense reward signals during JSON generation [26]. To guarantee structural validity, researchers have heavily explored constrained decoding and grammar-augmented generation. Ugare et al. presented SynCode, an efficient decoding algorithm utilizing an incremental parser to eliminate syntax errors in JSON and code generation without high computational overhead [27]. Chen et al. optimized this process further with Pre3, using deterministic pushdown automata to resolve the non-deterministic transitions typically associated with strict grammar constraints, significantly increasing system throughput [28]. As an alternative to altering the primary generation phase, Wang et al. introduced SLOT, a model-agnostic post-processing layer employing a compact, fine-tuned language model to reliably structure the unstructured textual outputs of larger models into precise JSON formats [29]. In specific domains, mapping natural language to API calls or extracting features requires strict structural adherence combined with semantic understanding. Chan et al. adapted LLMs for enterprise API integration by fine-tuning models on synthetic OpenAPI data and employing constrained

semantic decoding to map utterances to exact operation triplets [30]. In industrial settings, Escarda-Fernandez et al. transformed natural language queries into structured JSON for IoT devices using memory-optimized QLoRA fine-tuning, demonstrating that supervised tuning overcomes the limitations of standard zero-shot learning in highly specific domains [31]. Furthermore, Wang et al. demonstrated that grammar prompting, which leverages metalanguages like Backus Naur Form as an intermediate reasoning step, can significantly improve domain-specific generation when coupled with constrained decoding algorithms [32]. Exploring prompt and tuning efficiencies for structured tasks, frameworks like EaP optimize in-context example selection to enhance few-shot learning for complex e-commerce tasks without extensive manual prompt engineering [33]. Peng et al. showed that fine-tuning general-purpose LLMs with high-quality instruction datasets yields robust generalist models capable of generalizing to unseen domain-specific structures [34]. Finally, Wang et al. leveraged LoRA-inspired fine-tuning integrated with computer vision to guide customers in smart retail, proving that parameter-efficient tuning effectively bridges natural language understanding and structured task execution in resource-constrained environments [35]. These advancements highlight a critical distinction. Techniques such as schema-constrained decoding mathematically guarantee well-formed JSON, but they do not inherently improve the model's capacity to correctly understand the input text or accurately extract the correct entities. Consequently, a model utilizing only constrained decoding might produce perfectly formatted JSON that contains entirely incorrect data. For this reason, the present study isolates PEFT and prompt engineering as the primary comparison points for enhancing the underlying semantic accuracy and intent-mapping capabilities of LLMs, acknowledging that constrained decoding techniques could be applied orthogonally in production environments as a final structural safeguard.

Based on the cumulative evidence from previous studies, the goal of this work is to examine how adapter-based PEFT compares with advanced prompting strategies for structured parsing tasks. In particular, the study explores how different adaptation approaches influence structured output accuracy and reliability when generating JSON-formatted outputs. Guided by prior literature, the following hypotheses are considered:

- H1: PEFT methods may provide improvements in structured output generation performance compared to prompt-based adaptation strategies.
- H2: LoRA-based adapters may achieve higher structured extraction accuracy than other parameter-efficient tuning techniques such as IA^3.
- H3: Models of different sizes may respond differently to advanced prompt engineering strategies, suggesting that parameter-efficient adaptation could partially compensate for differences in model scale.
- H4: Prompt engineering alone may achieve high structured output accuracy, while PEFT may provide more consistent results across models.

This paper consists of six main sections. Section 1 introduces the research problem and motivation. Section 2 reviews related work on prompt engineering and parameter-efficient fine-tuning methods for large language models. Section 3 describes the experimental methodology, including dataset construction, model selection, adapter configurations, and evaluation protocols. Section 4 presents and analyzes the experimental results. Section 5 discusses the limitations of the study and outlines potential directions for future research. Finally, Section 6 summarizes the main findings and contributions of the study.

3. Methodology

This section describes the experimental framework designed to evaluate the trade-offs between PEFT and advanced ICL for structured information extraction. The primary objective is to measure performance, structural reliability, and generalization capability of

LLMs when converting unstructured natural language instructions into strictly formatted JSON objects. This methodology adapts the comparative benchmarking framework established by [3] and applies its evaluative principles to the specific challenges of a controlled structured parsing domain.

3.1. Task Definition

The experimental task focuses on Natural Language to Structured Command conversion, a core component of automated commerce and inventory systems. Given a free-form shopping instruction, the model must output a valid JSON object with the following schema:

```
{
  "action": "...",
  "quantity": "...",
  "product": "..."
}
```

The experimental task focuses on the "Natural Language to Structured Command" conversion, a critical component in automated inventory and e-commerce systems. To rigorously evaluate the effectiveness of both prompting and adapter-based fine-tuning, we constructed a controlled dataset comprising 12,000 annotated natural-language shopping instructions. To ensure high-quality and diverse data, samples were synthetically generated using a teacher-model paradigm.

Each instance requires the model to extract or infer three specific attributes and map them deterministically to a fixed JSON schema:

1. Action: Binary classification over {add, remove}.
2. Product: Named entity extraction identifying the canonical product name.
3. Quantity: Integer extraction with a default value of 1 when no explicit quantity is provided.

The mapping from natural language to JSON is deterministic, enabling objective evaluation of both semantic correctness and structural integrity.

3.2. Dataset Construction

The dataset consists of 12,000 synthetically generated but linguistically diverse shopping instructions, created in collaboration with industry partners to reflect realistic user interaction patterns. The dataset was synthetically generated to simulate natural language commands with variations in phrasing, quantity expressions, and product names, enabling evaluation of the models' ability to generalize across linguistic variations while producing deterministic structured outputs.

Although synthetically produced, the corpus incorporates substantial variation to prevent surface-form memorization.

3.3. Dataset Generation

To ensure a robust evaluation of the adaptation strategies under diverse linguistic conditions, the experimental dataset was significantly expanded from a foundational set of 1000 baseline samples used in our prior work. The dataset was scaled to 12,000 unique natural language commands using an automated data generation pipeline powered by state-of-the-art large language models, specifically Gemini 3.1 Pro and ChatGPT 5.2, acting as teacher models. These models were explicitly prompted to introduce high linguistic variance, incorporating a wide array of colloquialisms, paraphrasing patterns, and domain-specific slang terms. This synthetic expansion was crucial for simulating the noisy and

unpredictable nature of real-world user requests, thereby providing a more rigorous testbed for the structured semantic parsing task.

To guarantee the structural integrity and semantic accuracy of the generated dataset, a multi-stage quality control pipeline was established post-generation. First, algorithmic deduplication was executed to eliminate identical surface-form natural language strings. Second, an automated syntax validation script processed all generated pairs to ensure that every output strictly conformed to the required three-attribute JSON schema; any malformed, structurally incomplete, or unparsable instances were programmatically filtered out. Finally, the authors performed a manual spot-check on a randomized sample of 5% of the corpus ($n = 600$) to verify that the synthetic labels accurately aligned with the user intent and were free from semantic hallucinations. This combined verification protocol ensured a highly reliable and clean dataset for training and evaluation.

3.4. Model Configuration and Training Setup

The fine-tuning pipeline was strictly controlled to ensure reproducibility and optimal hardware utilization across both adapter methods. For the Low-Rank Adaptation (LoRA), the configuration was designed to maximize representational capacity across all attention and multi-layer perceptron (MLP) modules. The adaptation rank was set to $r = 32$ with a scaling factor $\alpha = 64$ and a dropout rate of 0.05. The target modules encompassed all primary projection matrices (`q_proj`, `v_proj`, `k_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`). LoRA training utilized a learning rate of 2×10^{-4}, a weight decay of 0.01, and enabled sequence packing with a per-device batch size of 8 and 4 gradient accumulation steps.

Conversely, the Infused Adapter by Inhibiting and Amplifying Inner Activations (IA^3) required a distinct optimization profile due to its reliance on scaling vectors rather than rank-decomposition matrices. For IA^3, the learning rate was increased to 1×10^{-3} and weight decay was completely disabled (0.0). Sequence packing was explicitly disabled, and `ddp_find_unused_parameters` was enabled alongside non-reentrant gradient checkpointing to prevent cross-contamination during Distributed Data Parallel (DDP) execution. The IA^3 per-device batch size was set to 4 with 2 gradient accumulation steps.

Despite these architectural differences, both adapters shared a robust baseline Supervised Fine-Tuning (SFT) framework optimized for NVIDIA A100 GPUs. Computations leveraged `bfloat16` precision and TensorFloat-32 (TF32) acceleration. The optimization process utilized a fused AdamW optimizer (`adamw_torch_fused`) with β parameters of $(0.9, 0.999)$ and a maximum gradient norm of 1.0. The models were trained for three epochs with a warmup ratio of 0.03 to stabilize initial gradients. A maximum sequence length of 1024 tokens was strictly enforced across all configurations to maintain memory efficiency and maximize training throughput.

Dataset Splitting for Generalization Assessment

To balance computational feasibility with sufficient diversity for adapter specialization, the 12,000 examples were partitioned into three disjoint subsets:

- Verb paraphrasing (e.g., add, insert, remove, delete, take out)
- Variable grammatical constructions
- Multi-word paraphrases and lexical substitutions

In Table 1, we present representative examples from the dataset, illustrating the range of linguistic variations and the corresponding JSON annotations.

Each example is paired with a normalized ground-truth JSON annotation containing the three labeled attributes.

Table 1. Dataset Examples and Corresponding JSON Annotations.

User_input	Action	Product	Quantity
Add 14 onions	add	onions	14
I need to remove 3 oranges	remove	oranges	3
I changed my mind about those socks, take 3 out.	remove	socks	3
Please grab 14 granola	add	granola	14

3.5. Data Splits

To ensure reliable model evaluation and prevent data leakage during training, the dataset was divided into three disjoint subsets for training, validation, and testing. The split was designed to provide sufficient data for adapter optimization while preserving a representative evaluation set for assessing model generalization.

The corpus was partitioned as follows:

- Training set: 8000 examples used for adapter parameter optimization during supervised fine-tuning.
- Validation set: 2000 examples used for monitoring model convergence, tuning training dynamics, and preventing overfitting.
- Test set: 2000 disjoint examples reserved exclusively for the final evaluation of model performance.

The validation and test sets remain completely unseen during adapter training. The test set additionally contains paraphrasing patterns and lexical combinations to evaluate the model's ability to generalize beyond the training distribution.

3.6. Model Selection

To analyze the effect of model scale and architectural variation, we evaluate a heterogeneous suite of LLMs across parameter sizes and training paradigms. We have selected three models that represent a range of scales and architectural designs, all of which are compatible with adapter-based fine-tuning, as shown in Table 2.

Table 2. Evaluated models and their parameter counts.

Name	Size
Llama 3.1	8B
Qwen 3	4B
Llama 3.2	3B

All models are evaluated under identical inference conditions with temperature set to 0 to eliminate stochastic variance.

3.7. Parameter-Efficient Fine-Tuning

For each base model, we compare three configurations:

1. Base Model (Prompting Only): No parameter updates; task performed via structured prompt engineering.
2. LoRA Adapter: Low-Rank Adaptation modules trained on the 8000-example training set.
3. IA^3 Adapter: Infused Adapter by Inhibiting and Amplifying Inner Activations trained under identical conditions.

Adapters are implemented using the PEFT framework within the Hugging Face Transformers ecosystem. Hyperparameters, tokenizer configuration, and preprocessing

steps are held constant across experiments to ensure fair comparison. No hyperparameter search is performed.

Training is conducted using multi-GPU Distributed Data Parallel (DDP).

3.8. Evaluation Protocol and Error Analysis

All configurations were evaluated on the same held-out test set of 2000 examples. Performance was measured using Precision, Recall, and F1 Score under strict exact-match criteria, with the F1 Score serving as the primary comparison metric.

A prediction was considered correct only if:

- The output is valid JSON and can be successfully parsed.
- All three fields (action, quantity, product) exactly match the ground truth.

If the model fails to produce valid JSON, the prediction is automatically counted as incorrect.

To rigorously assess model performance and systematically categorize failure modes, these strict criteria were implemented via a two-stage automated evaluation pipeline. The first stage evaluated the syntactic robustness of the generated outputs. By processing the raw model responses through a standard JSON parser, the system verified that each output constituted a well-formed, structurally valid object. Responses containing syntax errors, unclosed brackets, or invalid string formats were immediately classified as structural failures.

The second stage evaluated the semantic accuracy of the extracted entities. For all outputs that successfully passed the initial syntactic validation, the parsed key-value pairs were isolated. The corresponding attributes (specifically the designated action, extracted product item, and numerical quantity) were then compared against the ground-truth annotations in the dataset. A generation was classified as fully correct only if it achieved a deterministic match across all three schema fields. This strict evaluation protocol ensured that the models were penalized not only for formatting errors but also for subtle semantic hallucinations or extraction inaccuracies.

3.9. Adapter-Based Fine-Tuning (LoRA)

For the fine-tuning paradigm, we employed LoRA. This technique was chosen for its ability to maintain the underlying general knowledge of the base model while specializing its output structure for JSON parsing with minimal computational overhead.

The LoRA approach modifies the pre-trained weight matrices $W_0 \in \mathbb{R}^{d \times k}$ by adding a low-rank decomposition BA, where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$. During the forward pass, the output of the adapter is scaled by a factor of $\frac{\alpha}{r}$ to balance the retention of pre-trained information with the targeted domain adaptation. The forward pass for a given input x is thus represented as:

In this study, we targeted all linear modules within the transformer architecture to ensure that the adapter captured the structural and semantic constraints required for reliable JSON generation. The adaptation was applied to both the attention mechanism and the feed-forward network layers, specifically [36]:

- `q_proj`—Query projection layer that generates query vectors used to compute attention scores between tokens.
- `k_proj`—Key projection layer that produces key vectors used to determine the relevance of tokens during attention computation.
- `v_proj`—Value projection layer that encodes the information aggregated through the attention mechanism.
- `o_proj`—Output projection layer that combines the results of multiple attention heads and maps them back into the model's hidden representation space.

- `gate_proj`—Gating projection within the feed-forward network that regulates the activation flow in the nonlinear transformation stage.
- `up_proj`—Expansion projection layer that increases the dimensionality of hidden representations within the feed-forward block.
- `down_proj`—Compression projection layer that reduces the expanded representation back to the original hidden size.

The specific LoRA configuration utilized during fine-tuning included:

- Rank (r): 32. A higher rank was selected to provide sufficient parameter capacity for the complex semantic mapping required in structured parsing tasks.
- Alpha (α): 64. The scaling factor was set to $\alpha = 2r$, a heuristic commonly used to stabilize training and improve convergence when employing higher ranks.
- Dropout: 0.05, applied to the adapter layers as a regularization measure to mitigate overfitting.
- Precision and Compute: Training was executed on the SUPEK supercomputer, provided by the University Computing Centre (SRCE) of the CARNET infrastructure. This high-performance computing environment utilized NVIDIA A100 GPUs to support BFloat16 (`bf16`) mixed precision and TensorFloat-32 (`tf32`) optimizations, enabling accelerated matrix multiplications without the necessity for 4-bit quantization.

3.10. Infused Adapter by Inhibiting and Amplifying Inner Activations (IA^3)

While most fine-tuning methods focus on adding new structural layers or modifying a model's global weights, IA^3 adopts a minimalist approach by selectively inhibiting or amplifying existing internal signals. Instead of introducing heavy new components, IA^3 integrates three specific learnable vectors that act as signal modulators within the transformer's architecture [37].

3.10.1. The Signal Modulation Mechanism

The IA^3 method utilizes a mathematical operation known as a Hadamard product to precisely scale specific pieces of information as they flow through the system. The function of these adjustable modulators is divided into two primary roles:

- Attention Modulators (l_k and l_v): These vectors enable the model to prioritize specific parts of an input sentence. For example, in a shopping-cart command, these modulators can amplify the signal of a product name like "apples" while suppressing irrelevant filler words that do not contribute to the final structured output [38].
- Feed-Forward Modulator (l_{ff}): This vector modifies the signals within the model's memory center, specifically the feed-forward network. It assists the model in correctly categorizing the user's intent, such as distinguishing between an instruction to "add" an item or "remove" it from the digital cart [38].

3.10.2. Key Advantages for Practical Deployment

The IA^3 approach offers several distinct benefits for high-precision tasks under resource-constrained conditions:

- Extreme Parameter Efficiency: IA^3 represents a highly lightweight version of fine-tuning. It typically updates significantly fewer settings than other popular methods like LoRA, often modifying less than 0.01% of the total model parameters.
- Zero Performance Latency: Because these signal modulations can be mathematically merged into the original model weights after training, the customized model operates at the same speed as the original version. No additional computational overhead is introduced during the inference phase.

- Competitive Accuracy: Despite its small footprint, IA^3 has been demonstrated to match or even outperform significantly more complex fine-tuning methods on specialized reasoning and mathematical tasks.

By utilizing these precise signal modulators, IA^3 enables the model to achieve professional-grade precision in structured parsing without the heavy computational costs typically associated with larger fine-tuning strategies [37].

3.10.3. Optimization and Training Dynamics

The Supervised Fine-Tuning (SFT) phase was executed over 3 epochs with a maximum sequence length of 1024 tokens. To optimize memory footprint during backpropagation, gradient checkpointing was enabled. The training pipeline utilized a fused AdamW optimizer (`adamw_torch_fused`) with a peak learning rate of 2×10^{-4}, a weight decay of 0.01, and a warm-up ratio of 3% to ensure early-stage stability. An effective batch size of 32 was maintained by utilizing a per-device batch size of 8 combined with 4 gradient accumulation steps. The near-perfect performance achieved by LoRA-based adapters should be interpreted in the context of the controlled dataset and relatively constrained output schema used in this study. Because the task involves mapping natural language commands to a fixed JSON structure with a limited number of fields, adapter-based fine-tuning can rapidly learn task-specific patterns. While this result demonstrates the effectiveness of PEFT methods for structured extraction tasks, more complex schemas and real-world datasets may produce larger performance differences between adaptation strategies.

3.11. Prompt Engineering Strategies

To establish a baseline for ICL, we implemented three escalating levels of prompt complexity:

1. Minimal Instruction (Zero-Shot): A concise prompt defining the role of the model and the required JSON schema.
2. Extended Prompt: This strategy includes detailed edge-case instructions, emphasizing the default quantity logic and strict "no-prose" constraints to prevent conversational "chatter."
3. Few-Shot Prompting: Building upon the Extended Prompt, this version includes three representative examples of natural language inputs and their corresponding JSON outputs, providing a semantic and structural blueprint for the model.

3.12. Evaluation Metrics

The models were evaluated on two primary dimensions to assess both syntactic integrity and semantic accuracy.

3.12.1. JSON Validity Rate

Before measuring accuracy, we assess if the output is a "well-formed" JSON object. Any output that includes conversational filler, markdown errors, or fails standard library parsing (e.g., Python's `json.loads()`) is categorized as a failure. This metric directly evaluates the model's susceptibility to structural hallucination. JSON Validity Rate measures the proportion of generated outputs that conform to valid JSON syntax, ensuring that responses can be reliably parsed by downstream systems.

3.12.2. Structured F1 Score

For all valid JSON outputs, we calculate the F1 Score. This requires an exact match between the predicted value and the ground truth for each of the three fields (Action,

Product, Quantity). The field-level F1 Score is defined as the harmonic mean of precision (P) and recall (R) [39]:

$$\text{F1} = 2 \cdot \frac{P \cdot R}{P + R} \tag{1}$$

We report the macro-averaged F1 across all fields to provide a singular, robust measure of parsing performance.

3.12.3. Average Time per Test

To evaluate the practical deployment suitability of each adaptation strategy, we measure the Average Time per Test. This metric represents the computational efficiency and runtime cost associated with each approach during the inference phase. By recording the average duration (in seconds) required to process a single sample, we can quantify the trade-off between the high accuracy of complex prompting and the potential speed advantages of lightweight adapters.

4. Results and Discussion

This section evaluates the performance of three LLMs: Llama 3.1 8B, Llama 3.2 3B, and Qwen3 4B under multiple adaptation strategies, including minimal instruction prompting, extended prompting, few-shot prompting, and PEFT methods (IA^3 and LoRA). The evaluation was conducted on a dataset of 2000 samples, with performance measured using Exact Match Accuracy, F1 Score, Macro-Averaged F1, JSON Validity Rate for structured JSON output generation, and Average time per test. Table 3 summarizes the overall results across all evaluated approaches.

Table 3. Overall results across all evaluated approaches (Minimal—Minimal Instruction Prompting; Extended—Extended Instruction Prompting, Few-shot—Few-shot Instruction Prompting).

Model	Exact Match Accuracy	F1 Score	Macro-Averaged F1	JSON Validity Rate	Average Time per Test	Approach
Llama 3.1 8B	0.796	0.886	0.921	0.999	1.68	Minimal
Llama 3.2 3B	0.641	0.781	0.879	0.9945	1.2397	Minimal
Qwen3 4B	0.895	0.945	0.957	0.995	21.1431	Minimal
Llama 3.1 8B	0.874	0.932	0.949	1.000	1.7036	Extended
Llama 3.2 3B	0.691	0.817	0.889	0.9975	1.0425	Extended
Qwen3 4B	0.908	0.952	0.965	0.998	21.6727	Extended
Llama 3.1 8B	0.922	0.960	0.974	1.000	1.2096	Few-Shot
Llama 3.2 3B	0.864	0.927	0.958	1.000	0.7434	Few-Shot
Qwen3 4B	0.963	0.981	0.993	0.9985	18.4831	Few-Shot
Llama 3.1 8B	0.981	0.990	0.995	1.000	1.3842	LoRA
Llama 3.2 3B	0.985	0.992	0.989	1.000	1.1918	LoRA
Qwen3 4B	0.968	0.984	0.985	1.000	2.1289	LoRA
Llama 3.1 8B	0.796	0.886	0.921	0.997	1.5264	IA^3
Llama 3.2 3B	0.641	0.781	0.879	1.000	1.2222	IA^3
Qwen3 4B	0.895	0.945	0.957	1.000	11.6342	IA^3

The results show substantial variability across models and prompting strategies, reflecting the increased complexity of the evaluated dataset. For prompt-based approaches, F1 Scores range from 0.781 to 0.981, indicating that both model capacity and prompt design significantly influence performance.

Under minimal instruction prompting, Qwen3 4B achieves the strongest performance (F1 Score 0.945; Exact Match Accuracy 0.895). Llama 3.1 8B achieves moderate performance (F1 Score 0.886), while Llama 3.2 3B performs notably worse (F1 Score 0.781), suggesting

that smaller models are more sensitive to limited instruction. A visual comparison of model performance under minimal instruction prompting is presented in Figure 1.

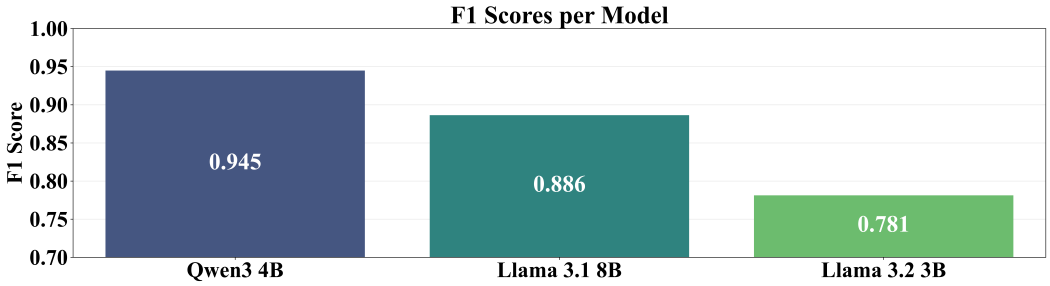


Figure 1. F1 Score under Minimal Instruction Prompting.

Extended prompting improves performance across all models. Llama 3.1 8B increases to an F1 Score of 0.932, Qwen3 4B to 0.952, and Llama 3.2 3B to 0.817. These results indicate that additional task specification enhances performance, although gains remain model-dependent. The corresponding results are illustrated in Figure 2.

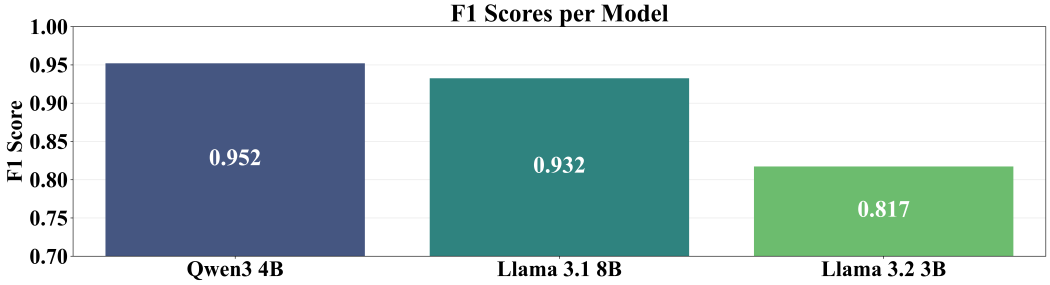


Figure 2. F1 Score under Extended Prompting.

Few-shot prompting yields the best overall performance among prompting strategies. Qwen3 4B achieves an F1 Score of 0.981 and Exact Match Accuracy of 0.963, while Llama 3.1 8B and Llama 3.2 3B reach F1 Scores of 0.960 and 0.927, respectively. These results are shown in Figure 3.

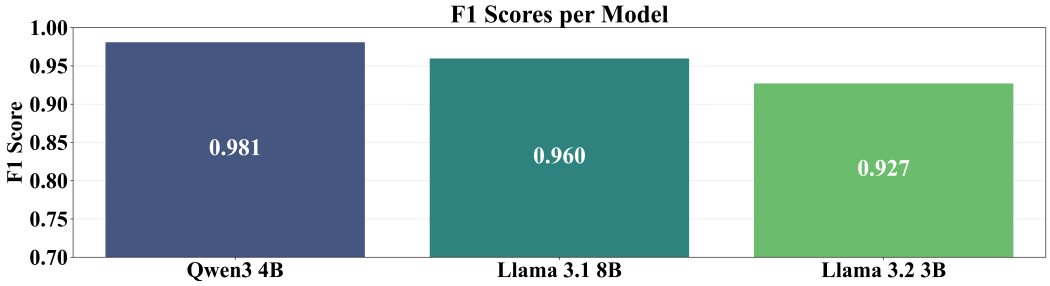


Figure 3. F1 Score under Few-Shot Prompting.

The observed improvements indicate that example-based conditioning is particularly effective in more complex settings, where explicit demonstrations help reduce ambiguity and improve alignment with the target schema.

Across all prompt-based strategies, Qwen3 4B consistently achieves the highest performance, followed by Llama 3.1 8B and Llama 3.2 3B, highlighting the impact of model architecture and scale.

4.1. Impact of Prompt Engineering Strategies

Prompt engineering strategies were evaluated to assess their influence on structured output generation.

Few-shot prompting achieves the highest performance across all models, followed by extended prompting, while minimal instruction prompting consistently performs worst. This ordering suggests that, for more complex inputs, providing structured examples is more effective than relying solely on concise instructions.

Despite differences in semantic accuracy, JSON Validity Rate remains consistently high (typically above 0.99) across all configurations, indicating that models reliably generate syntactically valid outputs even when prediction errors occur.

4.2. Performance of Parameter-Efficient Fine-Tuning Methods

To evaluate parameter-efficient adaptation, IA^3 and LoRA were applied to all models.

IA^3 produces results comparable to minimal instruction prompting. For example, Llama 3.1 8B and Llama 3.2 3B achieve identical F1 Scores to their minimal prompting configurations (0.886 and 0.781), while Qwen3 4B maintains an F1 Score of 0.945. These results are shown in Figure 4.

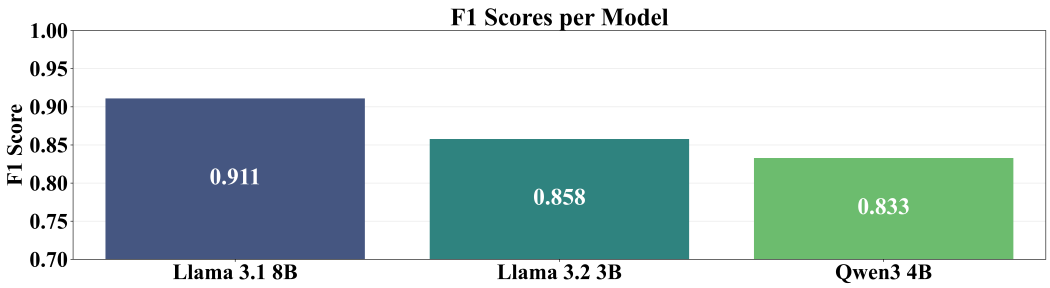


Figure 4. F1 Score under IA^3 Adaptation.

These findings suggest that, under the current configuration, IA^3 does not provide measurable improvements over prompt-based approaches. The performance disparity between LoRA and IA^3 is likely associated with their underlying architectural mechanisms and parameter capacities. IA^3 operates by scaling inner activations through learned vectors, which drastically limits the number of trainable parameters (yielding adapter sizes of merely 1–2 MB). While this vector-based signal modulation is highly efficient, it is hypothesized that under the evaluated configuration, it may lack the necessary expressive capacity to reliably capture the strict structural syntax and semantic mappings required for deterministic JSON generation. Conversely, LoRA introduces trainable rank-decomposition matrices into the self-attention mechanism, offering two orders of magnitude more trainable parameters (up to 320 MB). This increased capacity is assumed to enable LoRA to effectively override the model's pre-trained probabilistic generation tendencies and strictly adhere to the target JSON schema. However, because an explicit ablation study isolating parameter density from modulation mechanics was not performed, this interpretation must be framed strictly as a probable explanation under our current setup rather than as a definitive structural conclusion.

In contrast, LoRA achieves consistently strong performance across all evaluated models, though not perfectly. Llama 3.2 3B achieves the highest performance with an F1 Score of 0.992 and Exact Match Accuracy of 0.985, followed closely by Llama 3.1 8B (F1 Score of 0.990, Exact Match Accuracy of 0.981). Qwen3 4B achieves slightly lower results (F1 Score of 0.984, Exact Match Accuracy of 0.968), but still maintains high overall performance. All LoRA configurations achieve a JSON Validity Rate of 1.000, indicating fully consistent output formatting. The corresponding results are presented in Figure 5.

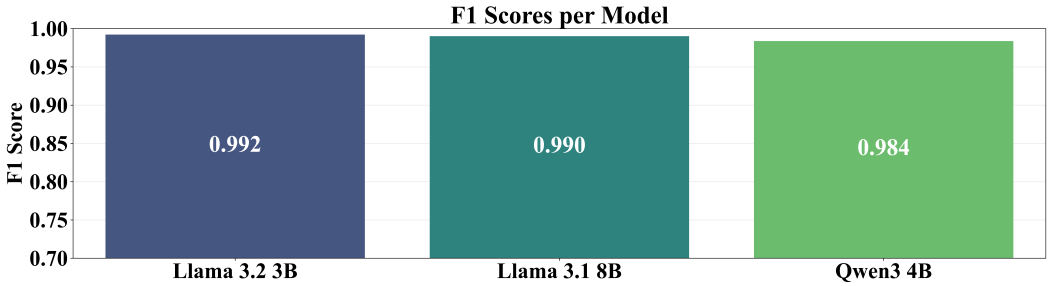


Figure 5. F1 Score under LoRA Adaptation.

In addition to accuracy improvements, LoRA demonstrates efficient inference performance, with average processing times ranging between 1.19 and 2.13 s per sample. Notably, LoRA-adapted models maintain comparable or lower inference times than several prompt-based configurations, suggesting that adapter-based specialization does not introduce significant runtime overhead.

The strong performance of LoRA in this setting can be attributed to its ability to introduce low-rank trainable matrices into attention layers, enabling the model to learn task-specific patterns without modifying the majority of base model parameters. This allows effective specialization for the target task while preserving the general capabilities of the pretrained model.

4.3. Comparison Between Prompt-Based and Adapter-Based Adaptation

Comparing prompt-based and adapter-based approaches reveals several key observations.

Prompt-based methods provide strong baseline performance, particularly with few-shot prompting. However, LoRA consistently achieves superior results, indicating that even lightweight fine-tuning can significantly improve reliability in structured generation tasks.

Model size effects are also reduced under LoRA. The Llama 3.2 3B model achieves performance comparable to larger models when fine-tuned, suggesting that PEFT can mitigate differences in model scale.

From a practical perspective, the choice between approaches depends on deployment constraints, as given in Table 4.

Table 4. Overview of prompt-based and adapter-based adaptation methods.

Prompt-Based Methods	Adapter-Based Methods
No training required	Higher reliability and accuracy
Lower setup complexity	Better control over output format
Flexible for multiple tasks	Improved performance for structured prediction tasks

Overall, prompt engineering provides a flexible baseline, while adapter-based tuning—particularly LoRA—offers improved consistency and accuracy.

4.4. Computational Cost: Training and Inference Analysis

In addition to accuracy-based metrics, this study evaluates the computational efficiency of different adaptation strategies using the average inference time per test sample. This metric provides an approximation of the runtime cost associated with each approach during deployment.

The results show notable differences in inference time across models and prompting strategies. Llama-based models demonstrate relatively low and consistent inference times, typically ranging between 0.7 and 1.7 s per sample across all prompting configurations. In contrast, Qwen3 4B exhibits significantly higher inference times, exceeding 18 s per

sample in few-shot prompting and over 21 s in minimal and extended prompting configurations. This discrepancy highlights that higher predictive performance does not necessarily correspond to lower computational cost. The disproportionately high inference latency of Qwen3 4B (despite its smaller parameter count compared to Llama 3.1 8B) represents a notable performance variation. Because dedicated micro-benchmarking and deep layer-wise profiling of the model's internal operations were not conducted within the scope of this study, the exact root causes of this latency spike remain empirically unverified. However, several architectural factors are hypothesized as potential contributing variables. These include hypothetical inefficiencies such as an unoptimized inference kernel for the specific hardware stack utilized, differences in tokenization efficiency when processing structured JSON strings, or a larger vocabulary size that could theoretically increase the computational overhead of the final softmax layer during sequential decoding. Consequently, these potential explanations should be interpreted strictly as plausible technical hypotheses rather than definitive findings, underscoring the necessity for further detailed hardware profiling in future research.

Prompt complexity also has a measurable impact on inference time. Few-shot prompting results in comparable or slightly reduced latency for Llama models, while Qwen3 4B maintains consistently high inference times regardless of prompt design. This suggests that the relationship between prompt length and latency is model-dependent and influenced by underlying architectural and implementation factors.

For adapter-based methods, both LoRA and IA^3 configurations yield inference times that are comparable to, or slightly lower than, those of the prompting approaches. This indicates that the addition of these lightweight adapter modules does not introduce any significant runtime overhead during inference.

Beyond inference cost, training efficiency and resource requirements must also be considered. As shown in Table 5, adapter training times range from approximately 5 to 16 min depending on the model and method. Although these durations appear modest, all experiments were conducted using high-performance hardware consisting of four NVIDIA A100-SXM4-40 GB GPUs and 64 CPU cores. Attempts to train adapters on less capable hardware were unsuccessful, indicating that such configurations may not be feasible in resource-constrained environments.

Table 5. Resource and Training Cost Comparison of Adapter Methods.

Adapter	Adapter	Training Time	Maximum GPU Memory Allocated	Adapter Size
Llama 3.1 8B	LoRA	13m 41s	17.7 GB	320.06 MB
Llama 3.2 3B	LoRA	11m 28s	17.7 GB	185.55 MB
Qwen3 4B	LoRA	16m 8s	17.7 GB	252.06 MB
Llama 3.1 8B	IA^3	6m 28s	17.86 GB	2.01 MB
Llama 3.2 3B	IA^3	5m 39s	17.86 GB	1.1 MB
Qwen3 4B	IA^3	7m 57s	17.86 GB	1.63 MB

The results further show that adapter size varies significantly across methods. IA^3 produces substantially smaller adapters (on the order of 1–2 MB), reflecting its higher parameter efficiency, whereas LoRA adapters are two orders of magnitude larger, reaching up to 320 MB. This difference is accompanied by longer training times for LoRA compared to IA^3, despite being trained on identical hardware.

From a practical perspective, these findings highlight an important trade-off between accuracy, latency, and resource requirements. While LoRA consistently achieves the highest performance across evaluated models, it also requires significantly greater computational resources for training. In contrast, prompting-based approaches require no additional

training and can be deployed on substantially less demanding hardware. However, it is essential to consider the long-term inference cost implications of prompt engineering at scale. While prompting incurs zero upfront training overhead, incorporating numerous contextual examples in few-shot scenarios significantly expands the input token length. In high-throughput environments, repeatedly computing attention across this extended context window for every user query leads to sustained compute costs and latency bottlenecks. In such cases, the one-time computational investment required to train a LoRA adapter can quickly amortize, resulting in a more computationally efficient system over time by allowing the base prompt to remain minimal.

Consequently, when high accuracy and output reliability are the primary objectives and sufficient computational resources are available, adapter-based methods are the preferred choice. Conversely, in scenarios with limited hardware availability or strict deployment constraints, prompt-based strategies offer a more accessible and cost-effective alternative.

4.5. Qualitative Error Analysis and Failure Modes

To better understand the limitations of the adaptation strategies, a comprehensive qualitative error analysis was conducted on the failed predictions. Across all evaluated configurations, structural parsing proved to be highly robust. Pure syntactic failures (cases where the model failed to generate a valid, parsable JSON object) accounted for less than 0.7% of all errors. This confirms that both advanced prompting and adapter-based fine-tuning effectively constrain the models to generate well-formed structural syntax. Consequently, the vast majority of errors (98.9%) were semantic mismatches, where the generated JSON was valid but the extracted key-value pairs deviated from the ground-truth schema.

The semantic mismatches were systematically categorized into two primary failure modes: action classification discrepancies and product normalization errors. A summarized typology of these recurrent failure patterns, including the observed frequency of errors across the top 15 most common mismatch categories, is presented in Table 6.

Table 6. Typology of Common Semantic Mismatches in Structured Parsing.

Category	Failure Type	Count	Expected	Output
Action	Literal Verb	195	add/remove	grab, put, cancel
Action	Semantic Flip	21	add	remove
Action	Case Issue	25	add	Add
Product	Over-extraction	51	bread	packs of bread
Product	Tokenization	82	oliveoil	olive oil
Product	Orthographic	67	raspberries	rasberries

Action Classification Discrepancies:As detailed in Table 6, the most prevalent source of error stemmed from the models' inability to map highly variable, colloquial verbs to the rigid, standardized action vocabulary defined by the schema. Under the "Literal Verb" failure type, models frequently extracted the literal verbs present in the synthetic prompts rather than translating them into the target categorical labels. Additionally, the analysis revealed instances of "Semantic Flipping", where models predicted the exact opposite target action for colloquial phrases like "I'll take," as well as "Case Issue" infractions, where models failed to strictly adhere to lowercase formatting constraints.

Product Normalization Errors: The second major category of failure involved entity boundary detection and product normalization. A recurring pattern of "Over-extraction" was observed, where models captured surrounding quantifier phrases or nouns alongside

the target entity. Furthermore, models struggled with "Tokenization" and spacing of compound words, frequently splitting merged dataset entities into separate linguistic tokens. Finally, "Orthographic" errors (minor spelling mistakes or missing characters) also contributed significantly to exact-match penalties.

Ultimately, these failure modes highlight a critical challenge in structured semantic parsing: while current adaptation strategies easily master rigid formatting constraints, robustly mapping high-variance, out-of-distribution natural language directly to strict database schemas remains a persistent bottleneck. This validates the necessity of evaluating these models against highly diverse, colloquial datasets.

4.6. Variability Across Models

The evaluation reveals consistent differences in model behavior.

Qwen3 4B achieves the best performance across prompt-based approaches, indicating strong instruction-following capabilities. Llama models show greater sensitivity to prompt design but benefit more from advanced prompting and fine-tuning.

Performance differences between 3B and 8B models are evident under prompting but diminish after adaptation, suggesting that task-specific tuning can reduce the impact of model scale.

Overall, both model architecture and adaptation strategy significantly influence structured output generation performance.

5. Limitations and Future Work

Although the experimental results demonstrate the effectiveness of both prompt engineering and PEFT for structured output generation, several limitations of the study should be acknowledged.

First, the evaluation was conducted on a synthetic dataset consisting of 2000 samples with a relatively simple schema, containing only three output attributes (action, product, and quantity). While this controlled setup enables clear evaluation of structured extraction performance, it does not fully capture the complexity of real-world information extraction tasks. In practical applications, schemas are often significantly larger and may include hierarchical structures, optional fields, or ambiguous entities, which can introduce additional challenges for language models.

Second, the study evaluates model performance using Exact Match Accuracy, F1 Score, Macro-Averaged F1, and JSON Validity Rate. These metrics effectively measure structured output correctness and formatting reliability, particularly for deterministic JSON generation tasks. Although baseline computational metrics—such as inference latency, memory consumption, and training time—were documented, they were measured exclusively on a high-performance GPU cluster. A systematic profiling of these costs under dynamic load conditions or across diverse, resource-constrained hardware architectures was not conducted, even though such factors heavily influence the practical selection of adaptation strategies.

Another limitation concerns the hardware requirements associated with adapter-based fine-tuning. Although PEFT methods significantly reduce the number of trainable parameters compared to full fine-tuning, training adapters for modern LLMs still requires substantial computational resources, including GPUs with sufficient memory capacity. Hardware constraints can restrict the feasible configuration of training parameters such as batch size, learning rate schedules, adapter rank, and training duration. As a result, the final performance of adapter-based methods may vary depending on the available computational environment and the chosen hyperparameter configuration.

Related to this, the study does not perform an extensive hyperparameter optimization for adapter training. Adapter-based methods such as LoRA and IA^3 can be sensitive to parameter choices, including rank values, scaling factors, dropout rates, and learning rates. Different configurations may produce varying results even when applied to the same base model and dataset. Consequently, the performance observed in this work should be interpreted as representative of a specific configuration rather than the absolute performance ceiling of each method.

Finally, while the models achieved very high performance on the evaluated task, particularly when using LoRA adapters, the dataset characteristics and limited schema complexity may contribute to this near-perfect performance. More complex structured extraction scenarios may reveal larger performance differences between adaptation strategies.

Future work could extend this study in several directions. One promising avenue involves evaluating the proposed approaches on larger and more complex datasets, including real-world information extraction tasks with richer schemas and nested JSON structures. Such datasets would provide a more challenging benchmark for assessing the robustness and generalization capabilities of both prompting and adapter-based approaches.

Another important direction is evaluating computational efficiency and resource requirements across different hardware tiers. While this study reports training time, GPU memory usage, and inference latency for a specific high-end configuration, future research could profile these factors on consumer-grade or edge devices to provide a more comprehensive cost-benefit comparison between prompt-based adaptation and PEFT in constrained environments.

Future studies could also explore hyperparameter sensitivity and optimization for adapter-based methods, analyzing how variations in adapter rank, learning rate, and training schedules influence performance. This would help identify stable configurations that provide consistent results across different hardware environments.

Finally, additional research may investigate robustness and generalization properties, including performance under paraphrased inputs, noisy text, or multilingual settings. Understanding how adaptation strategies behave under such conditions would provide deeper insights into the practical deployment of LLM-based structured information extraction systems.

6. Conclusions

This study investigated the effectiveness of prompt engineering strategies and parameter-efficient fine-tuning (PEFT) methods for adapting large language models to structured output generation tasks. Multiple prompting configurations (minimal instruction prompting, extended prompting, and few-shot prompting) were evaluated alongside two adapter-based techniques, IA^3 and LoRA, using three open-source LLMs of varying sizes. The models were assessed on their ability to transform natural language inputs into a predefined JSON schema.

The results demonstrate that modern LLMs provide strong baseline performance without parameter updates. Prompt-based approaches achieved consistently high accuracy across all evaluated models, with F1 Scores reaching up to 0.981. Among these strategies, few-shot prompting yielded the best overall performance in this experimental setup, indicating that explicit examples are particularly beneficial as task complexity increases.

Despite the effectiveness of prompt engineering, PEFT methods further improved performance and consistency. LoRA-based adapters achieved the highest overall results across all tested models, reaching near-perfect scores on all evaluation metrics. In contrast, IA^3 provided more modest improvements, with performance generally comparable to simpler prompting strategies. These findings confirm that lightweight parameter adaptation

can significantly enhance structured output reliability, even when starting from strong prompt-based baselines.

In addition to accuracy, the results highlight important computational trade-offs. Inference time varied substantially across models, with Llama-based models providing fast and consistent responses, while Qwen3 4B achieved higher accuracy at the cost of significantly increased latency. Furthermore, although adapter training times were relatively short (on the order of minutes), they required high-performance hardware, including multiple NVIDIA A100 GPUs and substantial CPU resources. This indicates that, while PEFT methods are parameter-efficient, their practical adoption depends on the availability of sufficient computational infrastructure.

The findings also show that PEFT can reduce performance differences between models of varying sizes. Notably, the smaller Llama 3.2 3B model, when adapted using LoRA, achieved performance comparable to larger models using prompting alone. This suggests that parameter-efficient adaptation can be an effective strategy for improving smaller models in resource-constrained deployment scenarios.

Overall, prompt engineering remains a flexible and low-cost approach that provides strong baseline performance without additional training. In contrast, adapter-based methods, particularly LoRA, offer superior accuracy and output consistency but introduce additional computational requirements during training. Consequently, the choice between prompting and PEFT should be guided by application-specific constraints, including accuracy requirements, latency tolerance, and available hardware resources, with no single approach universally optimal across all deployment scenarios.

To synthesize these findings within the scope of the evaluated controlled task, prompt engineering (specifically few-shot prompting) serves as an accessible baseline for scenarios where computational infrastructure is limited or rapid deployment is required, given its zero training overhead. Conversely, within the structural boundaries of our three-attribute parsing benchmark, LoRA-based fine-tuning demonstrates a highly reliable mechanism for ensuring schema compliance when sufficient baseline GPU infrastructure is available for offline optimization. This approach yields high output consistency and stable inference latencies under the tested constraints, indicating that localized adapter training can allow smaller models to match the precision of larger configurations on specific, narrow domains. While alternative methods like grammar-constrained decoding can mathematically enforce syntactic validity, these results suggest that within this experimental environment, LoRA adaptation provides a robust means of aligning the underlying semantic intent with fixed structural requirements. However, further validation across complex, deeply nested, or open-domain schemas is necessary before generalizing these advantages to broad production environments.

Author Contributions: Conceptualization, R.K., L.S. and S.B.Š.; methodology, L.S.; software, R.K.; validation, I.L. and S.B.Š.; formal analysis, R.K.; investigation, R.K. and L.S.; resources, R.K. and V.M.; data curation, R.K. and L.S.; writing—original draft preparation, R.K. and L.S.; writing—review and editing, S.B.Š., I.L., and V.M.; visualization, R.K.; supervision, S.B.Š. and I.L.; project administration, S.B.Š. and I.L.; funding acquisition, V.M. and I.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was (partially) supported by the EC Digital Europe Programme EDIH Adria 2.0 (101256325); SPIN projects IP.1.1.03.0120, IP.1.1.03.0158 and IP.1.1.03.0039; and NextGenerationEU University grants: uniri-iz-25-6, uniri-iz-25-220, IIP_010144, IIP_010136.

Data Availability Statement: The original data presented in this study are openly available in a public repository at https://github.com/FIPU-SPIN/llm-adapter-order-eval-fipu/, accessed on 17 May 2026.

Acknowledgments: This research was performed using the Advanced computing service provided by University of Zagreb University Computing Centre—SRCE.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LLMs	Large Language Models
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
LoRA	Low-Rank Adaptation
IA^3	Infused Adapter by Inhibiting and Amplifying Inner Activations
ICL	In-Context Learning
JSON	JavaScript Object Notation
SFT	Supervised Fine-Tuning
F1	F1 Score
GPU	Graphics Processing Unit
DDP	Distributed Data Parallel
MLP	Multi-Layer Perceptron
bf16	BFloat16 Precision Format
tf32	TensorFloat-32

References

1. Ling, C.; Zhao, X.; Lu, J.; Deng, C.; Zheng, C.; Wang, J.; Chowdhury, T.; Li, Y.; Cui, H.; Zhang, X.; et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *ACM Comput. Surv.* **2025**, *58*, 1–39. [CrossRef]
2. Han, Z.; Gao, C.; Liu, J.; Zhang, J.; Zhang, S.Q. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv* **2024**, arXiv:2403.14608.
3. Karlović, R.; Lorencin, I. Large language models as Retail Cart Assistants: A Prompt-Based Evaluation. In *Human Systems Engineering and Design (IHSED2025): Future Trends and Applications*; Applied Human Factors and Ergonomics International: Orlando, FL, USA, 2025; Volume 198.
4. Mundra, N.; Doddapaneni, S.; Dabre, R.; Kunchukuttan, A.; Puduppully, R.; Khapra, M.M. A comprehensive analysis of adapter efficiency. In Proceedings of the 7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD), Bangalore, India, 4–7 January 2024; pp. 136–154.
5. Hu, Z.; Wang, L.; Lan, Y.; Xu, W.; Lim, E.P.; Bing, L.; Xu, X.; Poria, S.; Lee, R. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 5254–5276.
6. Patil, R.; Khot, P.; Gudivada, V. Analyzing LLAMA3 Performance on Classification Task Using LoRA and QLoRA Techniques. *Appl. Sci.* **2025**, *15*, 3087. [CrossRef]
7. Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.M.; Chen, W.; et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mach. Intell.* **2023**, *5*, 220–235. [CrossRef]
8. Esmaeili, A.; Saberi, I.; Fard, F. Empirical studies of parameter efficient methods for large language models of code and knowledge transfer to r. *Empir. Softw. Eng.* **2026**, *31*, 30. [CrossRef]
9. Razuvayevskaya, O.; Wu, B.; Leite, J.A.; Heppell, F.; Srba, I.; Scarton, C.; Bontcheva, K.; Song, X. Comparison between parameter-efficient techniques and full fine-tuning: A case study on multilingual news article classification. *PLoS ONE* **2024**, *19*, e0301738. [CrossRef] [PubMed]
10. Shin, J.; Tang, C.; Mohati, T.; Nayebi, M.; Wang, S.; Hemmati, H. Prompt Engineering or Fine-Tuning: An Empirical Assessment of LLMs for Code. In *Proceedings of the 2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*; IEEE: New York, NY, USA, 2025; pp. 490–502.
11. Gong, M.; Deng, Y.; Qi, N.; Zou, Y.; Xue, Z.; Zi, Y. Structure-learnable adapter fine-tuning for parameter-efficient large language models. In *Proceedings of the International Conference on Artificial Intelligence and Computational Engineering (AICE 2025)*; IET: Bengaluru, India, 2025; Volume 2025, pp. 225–230.
12. Fang, X.; Ye, M. Towards Robust Parameter-Efficient Fine-Tuning for Federated Learning. In Proceedings of the Thirty-Ninth Annual Conference on Neural Information Processing Systems, San Diego, CA, USA, 2–7 December 2025.

13. Chen, S.; Gu, J.; Han, Z.; Ma, Y.; Torr, P.; Tresp, V. Benchmarking robustness of adaptation methods on pre-trained vision-language models. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 51758–51777.
14. Chen, L.C.; Weng, H.T.; Pardeshi, M.S.; Chen, C.M.; Sheu, R.K.; Pai, K.C. Evaluation of Prompt Engineering on the Performance of a Large Language Model in Document Information Extraction. *Electronics* **2025**, *14*, 2145. [CrossRef]
15. Zhu, K.; Wang, J.; Zhou, J.; Wang, Z.; Chen, H.; Wang, Y.; Yang, L.; Ye, W.; Zhang, Y.; Gong, N.; et al. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis, Salt Lake City, UT, USA, 14–18 October 2024; pp. 57–68.
16. Kim, J.; Mao, Y.; Hou, R.; Yu, H.; Liang, D.; Fung, P.; Wang, Q.; Feng, F.; Huang, L.; Khabsa, M. RoAST: Robustifying Language Models via Adversarial Perturbation with Selective Training. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, 6–10 December 2023; pp. 3412–3444.
17. Pei, A.; Yang, Z.; Zhu, S.; Cheng, R.; Jia, J. Selfprompt: Autonomously evaluating llm robustness via domain-constrained knowledge guidelines and refined adversarial prompts. In Proceedings of the 31st International Conference on Computational Linguistics, Abu Dhabi, United Arab Emirates, 19–24 January 2025; pp. 6840–6854.
18. Mu, L.; Chu, G.; Ni, L.; Sang, L.; Wu, Z.; Jin, P.; Zhang, Y. Robustness of Prompting: Enhancing Robustness of Large Language Models Against Prompting Attacks. *arXiv* **2025**, arXiv:2506.03627. [CrossRef]
19. Sajjadi Mohammadabadi, S.M.; Kara, B.C.; Eyupoglu, C.; Uzay, C.; Tosun, M.S.; Karakuş, O. A survey of large language models: Evolution, architectures, adaptation, benchmarking, applications, challenges, and societal implications. *Electronics* **2025**, *14*, 3580. [CrossRef]
20. Trad, F.; Chehab, A. Prompt engineering or fine-tuning? a case study on phishing detection with large language models. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 367–384. [CrossRef]
21. Pornprasit, C.; Tantithamthavorn, C. Fine-tuning and prompt engineering for large language models-based code review automation. *Inf. Softw. Technol.* **2024**, *175*, 107523. [CrossRef]
22. Wang, J.; Albarghouthi, A.; Sala, F. COSMOS: Predictable and Cost-Effective Adaptation of LLMs. *arXiv* **2025**, arXiv:2505.01449. [CrossRef]
23. Liu, M.X.; Liu, F.; Fiannaca, A.J.; Koo, T.; Dixon, L.; Terry, M.; Cai, C.J. "We need structured output": Towards user-centered constraints on large language model output. In Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, 11–16 May 2024; pp. 1–9.
24. Geng, S.; Cooper, H.; Moskal, M.; Jenkins, S.; Berman, J.; Ranchin, N.; West, R.; Horvitz, E.; Nori, H. Jsonschemabench: A rigorous benchmark of structured outputs for language models. *arXiv* **2025**, arXiv:2501.10868. [CrossRef]
25. Shorten, C.; Pierse, C.; Smith, T.B.; Cardenas, E.; Sharma, A.; Trengrove, J.; van Luijt, B. StructuredRAG: JSON Response Formatting with Large Language Models. *arXiv* **2024**, arXiv:2408.11061. [CrossRef]
26. Lu, Y.; Li, H.; Cong, X.; Zhang, Z.; Wu, Y.; Lin, Y.; Liu, Z.; Liu, F.; Sun, M. Learning to generate structured output with schema reinforcement learning. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vienna, Austria, 27 July–1 August 2025; pp. 4905–4918.
27. Ugare, S.; Suresh, T.; Kang, H.; Misailovic, S.; Singh, G. SynCode: LLM Generation with Grammar Augmentation. *arXiv* **2024** , arXiv:2403.01632. [CrossRef]
28. Chen, J.; Bai, S.; Wang, Z.; Wu, S.; Du, C.; Yang, H.; Gong, R.; Liu, S.; Wu, F.; Chen, G. Pre3: Enabling deterministic pushdown automata for faster structured LLM generation. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vienna, Austria, 27 July–1 August 2025; pp. 11253–11267.
29. Wang, D.Y.B.; Shen, Z.; Mishra, S.S.; Xu, Z.; Teng, Y.; Ding, H. Slot: Structuring the output of large language models. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, Suzhou, China, 4–9 November 2025; pp. 472–491.
30. Chan, R.; Mirylenka, K.; Gschwind, T.; Miksovic, C.; Scotton, P.; Toniato, E.; Labbi, A. Adapting LLMs for structured natural language API integration. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, Miami, FL, USA, 12–16 November 2024; pp. 991–1000.
31. Escarda-Fernández, M.; Botana, I.L.R.; Barro-Tojeiro, S.; Padrón-Cousillas, L.; Gonzalez-Vázquez, S.; Carreiro-Alonso, A.; Gómez-Area, P. LLMs on the Fly: Text-to-JSON for Custom API Calling. In Proceedings of the SEPLN (Projects and Demonstrations), Valladolid, Spain, 25–27 September 2024; pp. 130–135.
32. Wang, B.; Wang, Z.; Wang, X.; Cao, Y.; A Saurous, R.; Kim, Y. Grammar prompting for domain-specific language generation with large language models. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 65030–65055.
33. Zeng, J.; Dai, Z.; Liu, H.; Varshney, S.; Liu, Z.; Luo, C.; Li, Z.; He, Q.; Tang, X. Examples as the prompt: A scalable approach for efficient llm adaptation in e-commerce. In Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, Padua, Italy, 13–18 July 2025; pp. 4244–4248.
34. Peng, B.; Ling, X.; Chen, Z.; Sun, H.; Ning, X. eCeLLM: Generalizing Large Language Models for E-commerce from Large-scale, High-quality Instruction Data. *arXiv* **2024**, arXiv:2402.08831.

35. Wang, W.; Zhang, P.; Sun, C.; Feng, D. Smart customer service in unmanned retail store enhanced by large language model. *Sci. Rep.* **2024**, *14*, 19838. [CrossRef] [PubMed]
36. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. Lora: Low-rank adaptation of large language models. In proceedings of the ICLR 2022, Online, 25–29 April 2022; Volume 1, p. 3.
37. Liu, H.; Tam, D.; Muqeeth, M.; Mohta, J.; Huang, T.; Bansal, M.; Raffel, C. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. *arXiv* **2022**, arXiv:2205.05638.
38. Damana, A.A.; Hitesh, P. Innovative Adapter-Based Fine-Tuning and Streamlined Training Strategies for Text Summarization. In Proceedings of the 2024 International Conference on Electrical and Computer Engineering Researches (ICECER), Gaborone, Botswana, 4–6 December 2024; pp. 1–6. [CrossRef]
39. Wang, Z.; Pang, Y.; Lin, Y.; Zhu, X. Adaptable and reliable text classification using large language models. In *Proceedings of the 2024 IEEE International Conference on Data Mining Workshops (ICDMW)*; IEEE: New York, NY, USA, 2024; pp. 67–74.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.