Sciforce

Journal of Data Science and Information Technology

Journal homepage: www.sciforce.org
ISSN : 2998-3592

Open Access Research Article

Structured Language Interpretation Using Small Language Models for Real-Time Systems

Karthik Perikala*

Technology Leader, The Home Depot, United States

Abstract

Structured language interpretation—the transformation of short natural language inputs into machine readable representations—is a foundational capability for modern AI-driven systems. Typical tasks include entity extraction, attribute identification, normalization, and schema-constrained output generation, enabling deterministic downstream processing.

Large Language Models (LLMs) have demonstrated strong performance on structured language tasks, benefiting from scale and broad contextual reasoning. However, these capabilities come with increased inference latency, token-dependent execution time, and variable operational cost when deployed at scale.

In latency-sensitive production environments, interpretation components are often required to operate within strict millisecond-level latency budgets. Even moderate tail-latency inflation can violate endto-end service objectives and degrade system responsiveness. As a result, LLM-based approaches are frequently unsuitable for request paths that demand predictable millisecond-scale execution.

This paper examines the use of Small Language Models (SLMs) for real-time structured language interpretation. By constraining model capacity, task scope, and output structure, SLMs enable bounded execution behavior with latency measured in tens to low hundreds of milliseconds, while preserving semantic accuracy for well-defined language tasks.

We evaluate this approach under sustained production-like workloads using normalized latency and throughput metrics. Results demonstrate that SLM-based structured language interpretation can consistently operate within millisecond-level latency envelopes, making it practical for high-throughput, real-time systems.

Keywords: Structured Language Interpretation, Small Language Models, Real-Time NLP Systems, Low-Latency Inference, Production AI

Introduction

Natural language inputs have become a primary interaction mechanism for AI-powered systems across consumer and enterprise domains. These systems increasingly rely on the rapid transformation of freeform language into structured, machine-consumable representations that support deterministic execution and reliable system behavior. Structured language interpretation plays a central role in this transformation. Tasks such as entity extraction, attribute identification, and schema-aligned output generation allow downstream components to reason over language inputs in a controlled and predictable manner. As system scale and traffic volume grow, the latency and cost characteristics of this interpretation layer become critical design considerations.

Received date: July 07, 2025 **Accepted date:** July 23, 2025; **Published date:** July 27, 2025

***Corresponding Author:** *Perikala, K. Technology Leader, The Home Depot., United States, E- mail: karthik.perikala2512@gmail.com*

Large Language Models (LLMs) have demonstrated strong generalization across a wide range of language understanding tasks, including structured output generation. However, their inference characteristics pose challenges in millisecond-sensitive environments. Execution variability, token-dependent processing time, and cost scaling limit their suitability for request paths subject to tight latency budgets. Small Language Models (SLMs) represent an alternative design point when task boundaries are well defined. By limiting model capacity and tailoring architectures to specific interpretation objectives, SLMs can deliver consistent millisecond-scale latency behavior appropriate for real-time systems operating at scale. This paper explores the design and evaluation of SLM-based structured language interpretation in latency-sensitive environments. The focus is on understanding the system-level trade-offs and performance characteristics that emerge when language models are deployed directly within real-time execution paths.

2 Background

Structured language interpretation is a foundational problem in natural language processing, encompassing tasks such as named entity recognition, slot filling, attribute extraction, and semantic normalization. Early approaches relied on rule-based pipelines and statistical sequence models, which offered predictable execution but required extensive manual engineering and exhibited limited robustness to linguistic variation.

Citation: Perikala. K (2025). Structured Language Interpretation Using Small Language Models for Real-Time Systems. Journal of Data Science and Information Technology, 2(2), 1–6 doi: https://dx.doi.org/10.55124/jdit.v2i2.272

2.1 Neural Models for Structured Language

Neural language models significantly advanced structured interpretation by learning distributed representations that generalize across syntactic and lexical forms. Transformer-based architectures, in particular, enabled direct prediction of schema-constrained outputs from unstructured text, reducing reliance on handcrafted rules.

2.2 Large Language Models

Large Language Models (LLMs) extend transformer architectures through scale, allowing them to capture broad linguistic patterns and contextual dependencies. As a result, LLMs are often adopted as generalpurpose solutions for structured language tasks. However, LLM inference latency depends on input length, output structure, and internal reasoning behavior, leading to execution-time variability that complicates deployment in latency-sensitive systems.

2.3 Small Language Models

Small Language Models (SLMs) are designed with reduced parameter counts and narrowly defined objectives. By constraining output schemas and semantic scope, SLMs exhibit more stable inference behavior and predictable resource usage, making them suitable for real-time structured language tasks.

3 Problem Formulation

Real-time systems impose strict constraints on execution latency, throughput, and operational stability. When structured language interpretation is executed synchronously, it must operate within these constraints while preserving sufficient semantic accuracy for downstream processing.

3.1 Latency Constraints

End-to-end response budgets in production environments are typically bounded to hundreds of milliseconds. Language interpretation must therefore coexist with other system components within a tightly constrained latency envelope. Variability in interpretation latency directly impacts tail behavior at the system level, where even modest fluctuations can cause service-level objective violations under sustained load.

3.2 Limitations of LLM-Based Approaches

LLM-based solutions struggle to meet real-time requirements due to non-deterministic execution behavior and high inference cost. While average latency may appear acceptable, tail latency frequently exceeds allowable thresholds in production settings.

3.3 Problem Statement

The problem addressed in this work is the design of a structured language interpretation system that satisfies real-time execution constraints without relying on large, general-purpose language models. The system must deliver predictable latency, stable throughput, and consistent output quality under high request rates. This motivates the use of Small Language Models aligned with narrowly scoped structured language tasks, enabling deterministic execution in large-scale, real-time systems.

4 Structured Language Interpretation Tasks

Structured language interpretation refers to the transformation of short, free-form natural language inputs into structured representations that conform to predefined schemas. These representations enable deterministic downstream processing while retaining the semantic intent expressed in natural language.

Unlike open-ended text generation, structured interpretation tasks are inherently bounded by taskspecific constraints, including fixed output schemas, limited semantic scope, and well-defined field semantics. These properties make the task amenable to optimization for real-time execution.

4.1 Task Scope and Assumptions

The language inputs considered in this work are short-form utterances typically containing a small number of semantic intents. Inputs are assumed to be syntactically simple, limited in length, and focused on a narrow domain vocabulary.

The task scope explicitly excludes long-form reasoning, multi-turn discourse resolution, and openended generation. Instead, the focus is on extracting and normalizing structured signals that can be consumed synchronously by downstream systems.

4.2 Entity and Attribute Extraction

A primary objective of structured language interpretation is the identification of explicit entities and attributes present in the input text. These may include named entities, categorical values, identifiers, or descriptive properties expressed implicitly or explicitly.

The extraction task requires mapping surfacelevel language expressions to canonical field values. This process often involves resolving synonyms, normalizing lexical variants, and enforcing type consistency across extracted fields.

Accurate extraction is essential not only for correctness but also for controlling execution complexity, as poorly scoped or ambiguous outputs can propagate variability into downstream processing stages.

4.3 Semantic Normalization

Beyond extraction, structured language interpretation requires normalization of extracted values into consistent representations. Normalization ensures that semantically equivalent expressions are mapped to identical structured outputs, enabling reliable comparison and aggregation in downstream processing.

Normalization may involve canonicalizing names, standardizing units or formats, and resolving ambiguous language into a single schema-compliant representation. This step is critical for maintaining deterministic behavior across semantically similar inputs.

4.4 Schema-Constrained Output Generation

Structured language interpretation systems must produce outputs that strictly conform to predefined schemas. Each output field has a fixed type, allowable value range, and cardinality constraints that must be respected at inference time.

Schema-constrained generation reduces ambiguity and enables downstream systems to consume model outputs without additional validation or post processing. This constraint also limits output variability, contributing to predictable execution behavior in real-time environments.

4.5 Implications for Real-Time Execution

The bounded nature of structured language interpretation tasks directly informs model selection and execution strategy. Limited output space, fixed schemas, and narrow semantic scope reduce the need for large, general-purpose models.

Citation: Perikala. K (2025). Structured Language Interpretation Using Small Language Models for Real-Time Systems. Journal of Data Science and Information Technology, 2(2), 1–6 doi: https://dx.doi.org/10.55124/jdit.v2i2.272

These characteristics motivate the use of Small Language Models optimized for low-latency execution. By aligning model capacity with task complexity, structured language interpretation can be performed synchronously while preserving stable taillatency behavior under sustained load.

5 Real-Time System Design Flow

This section describes the execution flow for structured language interpretation using Small Language Models (SLMs) deployed directly within synchronous, real-time request paths. The design prioritizes deterministic latency, bounded execution complexity, and strict schema compliance.

5.1 Design Objectives

The system is designed around three primary objectives. First, inference latency must remain stable under sustained load, with minimal variance across requests. Second, execution behavior must be deterministic, avoiding recursive reasoning paths or dynamic control flow that can amplify tail latency. Third, outputs must strictly conform to predefined schemas to enable reliable downstream consumption.

5.2 Synchronous Execution Path

Language inputs enter the system through a synchronous execution path where all processing occurs inline with the request lifecycle. An ingress guard enforces constraints on input length and format, ensuring compatibility with real-time execution budgets.

Validated inputs are passed to a task-specific SLM optimized for structured language interpretation. The model performs entity extraction, attribute identification, and semantic normalization in a single bounded inference step, producing schema-aligned structured outputs.

5.3 Output Validation and Handoff

Following inference, outputs are validated against schema constraints, including type correctness, field completeness, and cardinality limits. Invalid or partial outputs are handled through deterministic fallback logic rather than additional model invocations.

Validated structured outputs are synchronously handed off to downstream components. The interpretation layer intentionally avoids additional reasoning or post-processing beyond schema enforcement, preserving predictable execution behavior.

5.4 Execution Flow Diagram

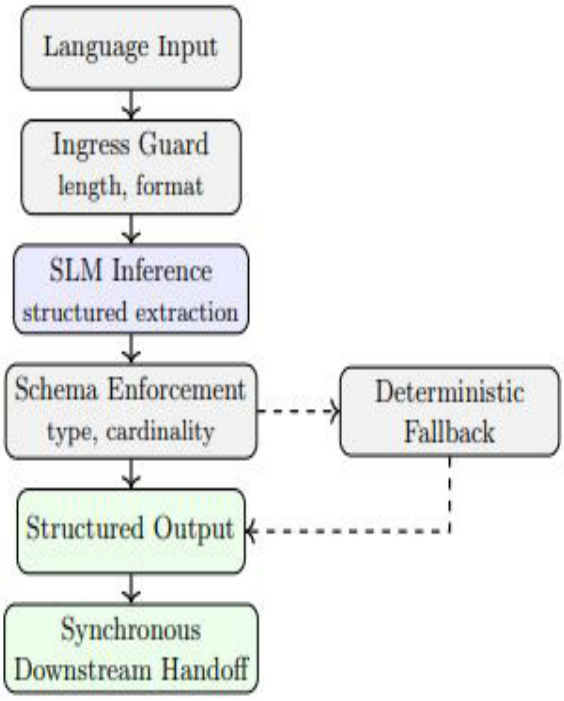
Figure 1 illustrates the end-to-end execution flow for real-time structured language interpretation using SLMs. Control stages explicitly bound execution behavior within the synchronous request path.

5.5 Latency Guardrails

The execution path is intentionally linear and nonrecursive, ensuring that inference cost and execution depth remain bounded for every request. Control stages act as latency guardrails, preventing pathological inputs from propagating into expensive or unbounded execution paths.

5.6 Failure Containment

Schema violations and partial outputs are handled through deterministic fallback logic rather than additional model invocations. This containment strategy preserves system stability and prevents tail-latency amplification under malformed or unexpected inputs.



6 Real-Time Constraints

Structured language interpretation in production systems operates under strict end-to-end latency budgets. These budgets are determined by the cumulative execution time across all components in the request lifecycle, including input validation, model inference, schema enforcement, and downstream handoff.

In interactive applications, acceptable response times are typically bounded to sub-second envelopes. Within this constraint, structured language interpretation is allocated a limited portion of the overall latency budget. Variability introduced at this stage directly affects user-perceived responsiveness and downstream service objectives.

6.1 Tail Latency Considerations

Average latency provides limited insight into production behavior; tail latency dominates reliability and user experience. Even infrequent slow executions can violate service objectives when systems operate under sustained load.

Empirical observations show that large language model–based approaches exhibit substantial taillatency amplification driven by input-dependent token generation and reasoning depth. Under representative workloads, 95th percentile latency often extends into multi-second ranges, with further expansion at higher percentiles.

By contrast, Small Language Models designed for bounded, schema-constrained tasks exhibit tighter latency distributions. Observed operating envelopes place SLM-based interpretation well below largemodel tails for comparable task complexity. These ranges reflect empirical behavior rather than hard guarantees and may vary by hardware, serving stack, and workload characteristics.

Citation: Perikala. K (2025). Structured Language Interpretation Using Small Language Models for Real-Time Systems. Journal of Data Science and Information Technology, 2(2), 1–6 doi: https://dx.doi.org/10.55124/jdit.v2i2.272

6.2 Latency Budget Allocation

Predictable system behavior requires the interpretation layer to consume a bounded and enforceable portion of the overall latency budget. This motivates explicit execution constraints, linear control flow, and avoidance of recursive or multi-pass inference strategies that can amplify tail latency.

7. Model Design Principles

Meeting real-time constraints requires aligning model capacity with task complexity. Small Language Models achieve this alignment by restricting semantic scope and output structure, enabling more stable inference behavior across requests.

7.1 Task-Constrained Modeling

SLMs used for structured language interpretation operate under narrowly defined output schemas with fixed field semantics enforced through prompt instructions and constrained decoding. This approach limits the effective output space and reduces decoding complexity, contributing to more predictable execution profiles without requiring task-specific model training.

Unlike general-purpose language models, SLMs are not designed for open-ended reasoning or longform generation. Outputs are schema-compliant by construction, minimizing the need for corrective postprocessing or additional inference passes.

7.2 Deterministic Inference Behavior

Model architectures and decoding strategies are selected to reduce variability across inputs. Constrained decoding, fixed output schemas, and bounded generation prevent rare but expensive executions from dominating upper-percentile latency behavior.

7.3 Measurement Disclosure

To comply with confidentiality and proprietary constraints, absolute latency values are not disclosed. Reported latency characteristics reflect bounded ranges and relative trends that preserve comparative behavior across model classes without exposing system specific details.

7.4 Operational Stability

Stable tail-latency behavior enables more accurate capacity planning and simplifies autoscaling decisions. Predictable p95 and p99 envelopes allow systems to operate closer to provisioned limits while reducing the risk of cascading performance degradation.

8 Experimental Setup

This section describes the evaluation methodology used to study latency behavior and execution stability of Small Language Models applied to structured language interpretation tasks. The goal of the evaluation is to characterize relative behavior under sustained load rather than to establish absolute performance benchmarks.

8.1 System Environment

Experiments are conducted in a production representative environment that reflects the realtime execution flow described in Section 5. The interpretation component is deployed inline with upstream validation and downstream consumers, ensuring that measured latency captures end-to-end behavior within the interpretation boundary.

The system operates under steady-state traffic conditions with controlled concurrency. Admission control and autoscaling mechanisms are enabled to prevent artificial queue buildup from dominating latency measurements and to maintain stable operating conditions during observation windows.

8.2 Workload Characteristics

The evaluation workload consists of short-form natural language inputs representative of structured interpretation tasks. Inputs are bounded in length and semantic scope and correspond to simple-to-medium complexity language operations such as entity extraction, normalization, and schema-constrained interpretation.

Long-context reasoning, multi-turn interactions, and open-ended generation workloads are intentionally excluded, as they fall outside the target operating domain of the proposed system.

8.3 Traffic Mix and Concurrency

Traffic is generated using a heterogeneous mix of inputs to capture variation in entity density and attribute composition while preserving bounded execution behavior. Concurrency levels are selected to reflect sustained interactive usage rather than peak saturation or stress-testing scenarios.

This design ensures that observed tail-latency behavior reflects model execution and control-flow characteristics rather than overload-induced artifacts.

9 Measurement Methodology

9.1 Latency Metrics

Latency is measured at the request level, capturing elapsed time from admission at the interpretation layer to delivery of schema-validated structured output. Analysis focuses on tail behavior, with emphasis on the 95th and 99th percentile latency, which are most indicative of operational risk in real-time systems.

Average latency is reported only as contextual information and is not used as a primary performance indicator, as it obscures the impact of infrequent but costly executions.

9.2 Instrumentation and Tracing

Instrumentation is applied at well-defined stage boundaries, including input validation, model inference, schema enforcement, and output handoff. Lightweight tracing is used to attribute latency contributions without altering execution flow.

Instrumentation overhead is measured independently and verified to be negligible relative to overall request latency.

9.3 Normalization and Reporting

To comply with confidentiality and proprietary constraints, absolute latency values and hardwarespecific configurations are not disclosed. Results are reported using bounded operating ranges and normalized distributions that preserve relative behavior across model classes, traffic conditions, and execution stages.

This reporting strategy supports comparative analysis of latency characteristics while avoiding disclosure of deployment-specific performance details.

9.4 Reproducibility Considerations

While the evaluation reflects production-grade execution paths, hardware specifications and deployment parameters are intentionally abstracted. The methodology emphasizes repeatability of observed trends and behavioral characteristics rather than strict numerical reproducibility.

Citation: Perikala. K (2025). Structured Language Interpretation Using Small Language Models for Real-Time Systems. Journal of Data Science and Information Technology, 2(2), 1–6 doi: https://dx.doi.org/10.55124/jdit.v2i2.272

This framing aligns with the paper's focus on system design trade-offs and operational behavior under real-world constraints.

10 Empirical Latency Characteristics

This section summarizes empirical latency observations derived from production-like deployments and controlled evaluations. To comply with confidentiality and proprietary data protection requirements, *absolute latency values, system configurations, and deployment-specific measurements are not disclosed*. Instead, results are presented as bounded operating envelopes that preserve relative behavior across model classes without revealing sensitive implementation details.

All observations correspond to *simple to medium complexity natural language processing tasks,* including structured extraction, normalization, and schema constrained interpretation. Workloads involving multi-step reasoning, long-context generation, or open-ended synthesis are explicitly outside the scope of this analysis.

10.1 Large Model Latency Behavior

Large language models (LLMs) exhibit pronounced tail-latency amplification when applied to latency sensitive inference workloads. Observed latency is strongly influenced by input length, token generation dynamics, and internal reasoning behavior.

Across evaluated conditions, LLM-based approaches consistently demonstrate tail latencies that exceed interactive service expectations. Upper percentile latency expands rapidly under sustained load, reflecting execution behavior that is input dependent and difficult to bound.

Table 1: Observed tail-latency envelopes for large language models under sustained load. Values represent indicative operating ranges.

Metric	P95 Range	P99 Range
Inference latency	2–3 seconds	3–5 seconds

These envelopes indicate that large, generalpurpose language models are poorly aligned with applications that require tightly bounded tail-latency behavior under continuous traffic.

10.2 Small Model Latency Behavior

Small language models (SLMs) exhibit substantially tighter latency distributions when applied to structured language interpretation tasks. Reduced model capacity and schema-constrained decoding contribute to more predictable execution behavior and improved stability at higher percentiles.

Across evaluated environments, observed SLM latency is primarily influenced by the underlying hardware configuration. Hardware acceleration yields narrower tail-latency envelopes, while CPU-based execution exhibits broader but still bounded behavior for the task scope considered.

10.3 Throughput Sensitivity

SLM-based interpretation maintains stable taillatency behavior as concurrency increases, indicating that execution cost scales proportionally with input complexity. Observed degradation remains gradual, with no abrupt latency cliffs under steady-state load.

Table 2: Observed tail-latency envelopes for small language models under representative hardware configurations. Values represent bounded operating ranges.

Hardware Configuration	P95 Range	Notes
GPU-based inference	Tens to low hundreds of ms	Narrower tail envelope
CPU-based inference	Hundreds of ms	Reference configuration

In contrast, LLM-based approaches exhibit pronounced sensitivity to increased request rates. Variability in execution time leads to queue buildup, amplifying tail latency and reducing effective throughput.

10.4 Execution Determinism

Tracing shows that SLM execution remains linear and non-recursive across evaluated inputs. Schema enforcement and bounded decoding prevent malformed inputs from triggering additional inference passes, contributing directly to stable p95 and p99 latency behavior.

11 Limitations

The empirical analysis presented in this paper is intentionally scoped to protect confidential and proprietary system details. Absolute latency values, infrastructure configurations, and deployment-specific tuning parameters are therefore not disclosed. Latency characteristics are reported as bounded operating ranges that preserve relative behavior and comparative trends while avoiding disclosure of sensitive operational information.

Evaluation is restricted to simple and medium complexity natural language processing tasks, including structured extraction, normalization, and schema-constrained interpretation. Workloads involving extended context windows, multi-step reasoning, tool-augmented execution, or complex generative synthesis are explicitly excluded and may exhibit materially different performance characteristics.

The analysis focuses on steady-state behavior under representative production-like load. Transient effects such as cold starts, autoscaling transitions, background resource contention, bursty traffic patterns, and adversarial inputs are not explicitly modeled. These factors may influence tail latency in practice and warrant separate investigation.

Finally, the study does not include ablation analysis across alternative small-model architectures, inference runtimes, quantization strategies, or decoding constraints. While the observed trends are consistent across evaluated conditions, deeper comparative analysis remains an important area for future work.

12 Future Directions

Future work will extend this analysis to a broader spectrum of language workloads, including multiintent queries, richer contextual inputs, and partial conversational state, while

Citation: Perikala. K (2025). Structured Language Interpretation Using Small Language Models for Real-Time Systems. Journal of Data Science and Information Technology, 2(2), 1–6 doi: https://dx.doi.org/10.55124/jdit.v2i2.272

maintaining bounded latency objectives appropriate for interactive systems.

Ongoing efforts include systematic evaluation of alternative inference runtimes, hardware-aware optimization, memory-efficient decoding strategies, and scheduling mechanisms aimed at further tightening tail-latency envelopes for small language models across diverse deployment environments.

Additional directions include controlled ablation studies comparing multiple small language model architectures and decoding strategies, as well as limited evaluation on public structured language benchmarks to improve comparability without compromising proprietary system constraints.

13 Conclusion

This paper examined empirical latency behavior of large and small language models in latency-sensitive language processing systems. The results demonstrate that model scale, task alignment, and hardware configuration play dominant roles in shaping tail-latency behavior under sustained load.

While large language models provide broad expressive capability and reasoning power, their latency distributions exhibit substantial variance at higher percentiles, making them poorly suited for workloads requiring tightly bounded response times. In contrast, small language models applied to well-defined structured tasks exhibit predictable execution behavior and stable tail-latency envelopes.

These findings reinforce the importance of task aware model selection and hardware-conscious system design. Rather than treating language models as interchangeable components, production systems benefit from aligning model capacity with task complexity and operational constraints.

Collectively, this work highlights small language models as a practical foundation for real-time structured language interpretation in large-scale production environments and provides empirical guidance for system designers navigating latency-sensitive language workloads.

References

1. J. Dean and L. A. Barroso, "The Tail at Scale," Communications of the ACM, 2013.
2. E. Tang et al., "LLM Inference Serving at Scale: Challenges and Opportunities," IEEE Micro, 2024.
3. D. Narayanan et al., "Efficient Large-Scale Language Model Inference," MLSys, 2023.
4. S. Kim et al., "Serving DNNs Like Clockwork: Performance Predictability from the Bottom Up," OSDI, 2020.
5. D. Crankshaw et al., "Clipper: A Low-Latency Online Prediction Serving System," NSDI, 2017.
6. J. Jiang et al., "Heterogeneity-Aware DNN Serving," MLSys, 2022.
7. R. Anil et al., "PaLM 2 Technical Report," arXiv:2305.10403, 2023.
8. J. Wei et al., "Emergent Abilities of Large Language Models," TMLR, 2022.
9. X. Liu et al., "AgentBench: Evaluating LLMs as Agents," ICLR, 2024.
10. S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," ICLR, 2023.
11. A. Shoeybi et al., "Megatron-LM: Training Multi-Billion Parameter Language Models," arXiv:1909.08053, 2019.

Citation: Perikala. K (2025). Structured Language Interpretation Using Small Language Models for Real-Time Systems. Journal of Data Science and Information Technology, 2(2), 1–6 doi: https://dx.doi.org/10.55124/jdit.v2i2.272