Enhancing Inference in Small Language Models with Model-Agnostic Techniques

Yisel Clavel-Quintero[1,*], Dionis López-Ramos[2], Derek-Naharai Herrera-Domínguez[3] and Edivaldo-Armindo Machado-Gomes[4]

[1]University of Holguín (UHO), Holguín, Cuba
[2]University of Oriente (UO), Santiago de Cuba, Cuba
[3]University of Computer Science (UCI), Havana, Cuba
[4]Independent, Valencia, Spain

Abstract

Structured information extraction from unstructured text remains a critical challenge in natural language processing. Large Language Models offer solutions; nevertheless, their computational cost and latency limit their deployment. Small Language Models provide an efficient alternative, yet their reduced parameter count often leads to lower precision and a higher incidence of hallucinations. This paper explores three model-agnostic techniques to enhance inference in small language models for the GenSIE 2026 task: Prompt Engineering, Domain-Aware Retrieval-Augmented Generation, and Grammar-Constrained Decoding. Evaluated on the Llama, Salamandra, Gemma, and Qwen families using the GenSIE starter and test datasets, the RAG-Domain-GloVe pipeline achieved the highest performance on the starter dataset with a micro-F1 of 0.72 on Qwen 2.5 14B, while GCD delivered the best results on the test set with a micro-F1 of 0.56 on Gemma 4 E4B and a gap-closed of 25.22%. The Prompt-Eng (HFE) pipeline outperformed all other techniques on the test set with Qwen 3 14B (F1 = 0.61), closing a gap of 12.06%, confirming the effectiveness of structured prompting for larger models. All three pipelines ran efficiently on standard consumer hardware, demonstrating that high-quality structured extraction is feasible without reliance on costly cloud APIs. Overall, the results demonstrate that model-agnostic prompting structures, domain-aware retrieval augmentation and constrained generation methods can significantly boost the precision and recall of small models, making them highly viable for production environments with limited hardware resources.

Keywords

Small Language Models, Prompt Engineering, Retrieval-Augmented Generation, Grammar-Constrained Decoding, Information Extraction

1. Introduction

The proliferation of Large Language Models (LLM) has transformed natural language processing (NLP), enabling complex tasks like zero-shot information extraction, where the extraction of structured information from unstructured text represents one of the most critical bottlenecks. While LLM are powerful, their outputs are often unpredictable. Most existing solutions attempt to correct erroneous outputs after generation using parsing, regular expressions, or brittle code that breaks easily.

However, the computational and memory footprint of frontier models (with tens to hundreds of billions of parameters) remains a significant barrier for local, real-time, or resource-constrained applications [1]. In response, the field has seen a surge in Small Language Models (SLM) with 1B to 14B parameters, which promise efficiency without a catastrophic drop in capabilities. In this scenario, model-agnostic inference techniques are needed to improve the quality of model outputs, bringing them closer to those of large models.

IberLEF 2026, September 2026, León, Spain
*Corresponding author.
yclavelq@uho.edu.cu (Y. Clavel-Quintero); dionis@uo.edu.cu (D. López-Ramos); derekd@estudiantes.uci.cu (D. Herrera-Domínguez); edivaldogomes6671@gmail.com (E. Machado-Gomes)
0000-0002-0174-4031 (Y. Clavel-Quintero); 0000-0001-9217-0696 (D. López-Ramos); 0009-0001-4718-5730 (E. Machado-Gomes)

The GenSIE (General-purpose Schema-guided Information Extraction) 2026 task, part of IberLEF [2, 3], is designed to evaluate these smaller, open-weight models. The challenge requires systems to extract nested JSON structures from Spanish text across diverse domains, with the schema provided only at inference time.

This research, presented by the JSONautas team, investigates three model-agnostic enhancement techniques to improve SLM performance on this task: (1) a structured Prompt Engineering approach, (2) a Domain-Aware Retrieval-Augmented Generation (RAG) approach, and (3) a Grammar-Constrained Decoding (GCD) approach. We evaluate these techniques across the Llama 3.2, Salamandra 7B, Qwen 2.5 14B, Gemma 4 E4B, and Qwen 3 14B models. The paper is structured as follows: Section 2 presents the state of the art and related work; Section 3 describes the proposed pipelines and the techniques evaluated; Section 4 reports the experimental results and analysis; and Section 5 concludes the work and offers recommendations.

2. State of the Art

This section presents a state-of-the-art (SOTA) review of three techniques: Prompt Engineering, Retrieval-Augmented Generation, and Grammar-Constrained Decoding, oriented toward structured information extraction. We analyze the main strategies identified in the recent literature.

2.1. Prompt engineering

Recent research converges on several principles for guiding LLM, particularly smaller ones, toward reliable structured outputs, i.e., two-shot anchoring, semantic isolation, technological agnosticism, and the impact of formatting style. In order for a prompt engineering technique to be effective in heterogeneous environments, it must not depend on architecture-specific tokens, but rather on the logical structure of the message. Using formats such as JSON or XML provides portability across different providers and model versions, reducing the technical debt of extraction systems [4, 1].

Mondal et al. [5] empirically demonstrate that the number of in-context examples is a determining factor, showing that two-shot anchoring is often sufficient, with diminishing returns beyond a few examples. Providing exactly one or two examples is sufficient to reliably guide the model toward predefined structures, while adding more examples tends to degrade response quality and increase computational cost without appreciable marginal benefits. This phenomenon holds consistently across a wide variety of datasets, models, and output structures.

For SLM, Mayilvaghanan et al. [6] propose PROPEL (Prompt Optimization with Expert Priors for LLM), an iterative optimization system based on a three-agent loop. Results show improvements of up to 9% on Llama-3.2 1B and 3B models, demonstrating that affirmative directives and semantic isolation via clear delimiters are more effective than complex few-shot chains for small models. In addition, [4] presents another semantic isolation and model-agnostic approach, SLOT (Structured LLM Output Transformer), which functions as a lightweight post-processing layer, transforming unstructured LLM outputs into precise structured formats without requiring retraining.

Elnashar et al. [1] demonstrated that prompt formatting style critically impacts performance. Their findings reveal key trade-offs: JSON schemas yield high precision for complex and nested data, YAML format keeps a balance between human readability and computational efficiency, hybrid CSV/Prefix files show maximum token efficiency for flat structures, and XML semantic zones reduce instruction-data confusion. The authors conclude that structures with explicit delimiters consistently reduce syntax errors in model output.

2.2. Retrieval-Augmented Generation

The current SOTA for running RAG with SLM focuses on specialization and optimized pipelines with modular and intelligent orchestration. Rather than relying on a single best model, the prevailing approach consists of selecting the right combination of small, specialized components for embedding,

retrieval, and generation to achieve a balance between high accuracy and low computational cost [7, 8]. Key techniques include hybrid retrieval (dense and sparse), cross-encoder reranking, and agentic workflows such as multi-agent RAG.

Incorporating domain-specific lexicons and term glossaries in the vector store is considered a state-of-the-art best practice for bridging the "vocabulary gap" between user queries and knowledge base terminology [9, 10]. These specialized knowledge files are most effective when integrated into different stages of the RAG pipeline: query augmentation expands technical terms with definitions, improving retrieval accuracy; retrieval enhancement combines dense vector embeddings with sparse lexical search (BM25) to capture both semantic meaning and exact terminology, further supercharged by term-level RAG; metadata filtering tags document chunks with glossary terms during indexing and pre-filters the search space at query time to increase precision; chunking treats each term-definition pair as an independent atomic chunk to prevent arbitrary splitting; and prompt enrichment injects retrieved glossary definitions directly to the LLM's prompt as explicit context [7, 10, 8, 11].

For the embedding and retrieval component, specialized small embedding models such as `langcache-embed-v3-small` (∼20M parameters) are designed for semantic caching and intent detection: when a new query shares the same intent as a previous one, cached answers can be returned instantly, saving significant computation [12, 7]. In terms of generation and reasoning, agentic frameworks such as PolyRAG and Haystack enable multi-step reasoning with SLMs by chaining tools like SQL queries and document retrievers, while FrugalRAG uses reinforcement learning to teach a 7B model when to stop searching, cutting retrieval costs by up to 50% without sacrificing accuracy on complex multi-hop questions [13, 14, 9].

Lightweight parameter-efficient fine-tuning methods like QLoRA can adapt small base models such as Llama 3.2 3B to excel at specific tasks like structured data extraction, allowing them to match or even surpass the performance of much larger models on targeted tasks [8]. Performance evaluations demonstrate that models including Llama 3.1 8B, Llama 3.2 3B, and Qwen3 8B have demonstrated extraction accuracy and coverage comparable to larger proprietary models, although nuanced differences may remain in areas such as summary quality when compared to a GPT-4 baseline, while the primary advantage is efficiency and cost reduction [15, 8, 14].

For processing structured JSON data, a new and sometimes more effective approach bypasses the traditional "embedding + vector database" pipeline entirely, employing methods that enable LLM to directly parse and reason over the structure of JSON files [16, 17]. Specialized small models for JSON extraction from raw text include NuExtract-2.0-4B [16], which uses a JSON template where the user describes the structure and the model fills it; NuExtract-tiny-v1.5, a 0.5B lightweight version based on Qwen2.5 ideal for resource-constrained environments; Liquid AI's LFM2-1.2B-Extract [17], optimized for local multi-language data extraction; and SmolLM2-360M-Instruct-Text-2-JSON [18], an ultra-small model fine-tuned for extracting information into structured JSON. Vectorless and hierarchical reasoning tools such as `ragzero-core` challenge the traditional RAG stack by parsing documents like JSON into a hierarchical tree and using an LLM to select the most relevant sections for a query, which simplifies deployment, reduces infrastructure needs, and is particularly effective for structured data [19].

2.3. Grammar-Constrained Decoding

Constraint sampling guides text generation toward specific formatting constraints by filtering logits to retain only valid tokens [20]. However, while effective for structured formats such as JSON, YAML, regex, and CSV, implementing the required grammars is a non-trivial task. Since each output format requires its own grammar, constraint-based sampling is less generalizable. Moreover, grammar verification can increase generation latency. Also, while equivalent grammars produce identical token masks, they impose vastly different computational costs.

Recent advances such as Outlines [21] provide a balance of speed and flexibility, achieving a 97% success rate and a low hallucination rate (0.4%) in few-shot scenarios by ensuring that the generated output conforms to a formal grammar. This makes Outlines a robust solution for most production settings, offering high success rates, balanced performance, and low error rates, particularly when

representative few-shot examples are available.

Research by [22] established a formal invariance theorem for GCD, introducing the Structural Ambiguity Cost (SAC) which proves that right-recursive grammars are significantly more efficient than left-recursive ones. New research like LAVE (Lookahead-then-Verify) is adapting constrained decoding for Diffusion LLM. This approach ensures reliability without sacrificing the parallel generation capabilities of these emerging architectures [23].

3. Pipelines proposed

The following three model-agnostic pipelines[1] were designed and evaluated on the GenSIE datasets, including data from different domains, such as cultural, environmental, general, legal, medical, research, STEM, and technical. The starter dataset was tested using four local SLM (Llama 3.2 1B/3B, Salamandra 7B, and Qwen2.5 14B), and the test dataset was benchmarked using two SML (Gema4 E4B and Qwen3 14B). The task imposed a time limit of 60 seconds per extraction, as defined by the organizers.

3.1. Datasets

We use the starter and test datasets curated and facilitated by the GenSIE task organizers[2]. The starter dataset has 40 silver-standard instances. This dataset spans four distinct domains (i.e., cultural, legal, medical, and technical) with examples drawn from movie reviews, constitutional legislation, drug safety leaflets, and software documentation. Each instance is a single JSON file (input_text, instruction, target_schema, output, metadata) comprising a source text in Spanish, a natural language instruction also in Spanish, and a complex target schema in English that ranges from literal extraction (L1) to hierarchical entity grouping (L5), logical inference (L8), and adversarial null handling (L9). The intermediate complexity levels (L2–L4) include tasks such as single-slot value extraction, multi-slot record extraction, and flat list extraction. This diversity challenges models to perform nested information extraction, opinion distillation, binary reasoning, and semantic mapping. The starter dataset serves as a development and validation resource, enabling participants to benchmark the schema-guided agents under the Flattened Schema Scoring (micro-F1) metric before final submission.

An additional development set with 150 silver instances was released for experimentation. Although we did not evaluate any metrics on this set, it proved useful for identifying additional domains. This dataset extends the domain coverage as follows: Cultural (literature, monuments, movie_reviews); Environmental (ecology); General (disasters); Legal (contracts, judicial, legislation); Lifestyle (recipes); Medical (diseases, drug_safety, health_news); STEM (astronomy_detailed); and Technical (software).

The test dataset comprises 145 instances, also curated by the GenSIE organizers, and follows the same multi-domain and multi-complexity schema structure as the starter set. This dataset covers the eight task domains: cultural, environmental, general, legal, medical, research, STEM, and technical, with subdomains including movie reviews, climate reports, news articles, constitutional legislation, clinical trials, scientific abstracts, mathematical problems, and software documentation. Each instance has the same shape as the starter set, and the output field provides the gold annotation to enable local scoring. These instances are held out from the public leaderboard and serve as the official evaluation benchmark for the shared task. Unlike the starter dataset, the test set is designed to assess generalization capabilities under the same Flattened Schema Scoring metric, with results reported on a blind test split to ensure fair comparison across participating systems. The test set was released with its gold labels to allow participants to evaluate locally and self-assess their systems for transparent evaluation.

3.2. Pipeline 1: Prompt engineering ("`prompt_eng`")

This zero-shot pipeline focuses on instructional brevity and structural coherence. Drawing on the principles established by [6] and [1], it employs a flat, affirmative system prompt in Spanish with explicit

[1]Our implementations are publicly available at https://github.com/yiselclavel/GenSIE2026-JSONautas.
[2]All the datasets are publicly available at https://github.com/gia-uh/gensie.

delimiters and no in-context examples.

A first variant of the high-fidelity extraction (HFE) approach (`system_prompt_gcd`) defines a professional role, isolates the target schema, and enforces a strict "null" literal for missing information. A second variant (`prompt_eng_gcd_xml`) employs XML semantic zones (`<context>`, `<data>`, `<task>`, `<constraints>`, `<output_format>`) to prevent instruction-data confusion. Both variants require the output to be pure JSON, with no Markdown blocks. This pipeline is optimal for low-latency SLM (1B-8B) where maximum token efficiency is desired. The prompt structure is defined as follows:

- `<context>`: Defines the professional role and application scenario for the model.
- `<data>`: Isolates the target schema and the source text to be processed.
- `<task>`: Declares the transformation action unambiguously.
- `<constraints>`: Establishes barriers against hallucinations, including the mandatory `"null"` literal rule.
- `<output_format>`: Specifies the final output format (pure JSON, no Markdown blocks).

3.3. Pipeline 2: Domain-aware RAG ("rag")

This pipeline enhances the prompt with domain-specific knowledge retrieved from a pre-indexed glossary. The implementations are found in `rag_*_agent.py`. The best-performing variant uses a lightweight GloVe [24] embedding model with six billion tokens and a 50-dimensional vector space (6B.50d), offering an alternative to computationally expensive dense retrievers based on transformer architectures.

We added a small amount of publicly available data for each domain and subdomain. The datasets were cleaned, and only the plain text was extracted. The files with a format distinct from CSV were converted to TXT. Due to availability constraints, the information is mostly in English, with one source in Spanish (marked as *es*). The domains and subdomains available are as follows (format: "domain: subdomain"):

- cultural: literature[3], monuments (es)[4], movie_reviews[5]
- environmental: ecology[6]
- legal: judicial[7]
- lifestyle: recipes[8]
- medical: drug_safety[9]
- technical: software[10]

The main workflow consists of:

1. Domain detection: This step extracts the domain and subdomain from the task metadata or task ID. When a subdomain exists, it is loaded; otherwise, the model retrieves information from other available subdomains.
2. Vector retrieval: Uses cosine similarity to find the top-k (k=3) most relevant glossary chunks from a pre-processed domain-specific corpus.
3. Context injection: The retrieved glossary definitions are injected as a "GLOSARIO DE REFERENCIA" section in the user prompt.

[3] https://www.kaggle.com/datasets/thedevastator/comprehensive-literary-greats-dataset
[4] https://object.pouta.csc.fi/OPUS-ELRC-2638-monumentos_2007/v1/mono/es.txt.gz
[5] https://zenodo.org/records/14203180
[6] https://plos.figshare.com/articles/dataset/_Examples_of_variables_potentially_suitable_for_assessing_the_severity_of_environmental_degradation_under_criterion_C_/700262?file=1056903
[7] https://archive.ics.uci.edu/dataset/239/legal+case+reports
[8] https://www.kaggle.com/datasets/thedevastator/better-recipes-for-a-better-life
[9] https://archive.ics.uci.edu/dataset/461/drug+review+dataset+druglib+com
[10] https://huggingface.co/datasets/JasonWong0516/software-engineering-datasets,https://www.kaggle.com/datasets/syedmharis/software-engineering-interview-questions-dataset

4. Structured generation: An OpenAI-compatible client calls the local model with the enriched prompt, enforcing the response as a JSON schema.

We explore different architectural choices for this pipeline in order to enhance small language models with external knowledge. Next, we detail and compare each variant evaluated.

1. RAGAgent - Baseline GloVe-based retrieval: The architecture consists of three primary components: a GloVeEmbeddings class responsible for loading and managing pre-trained word vectors from the standard GloVe 6B corpus; a DocumentStore that maintains an in-memory collection of documents, their corresponding embeddings, and associated metadata; and the RAGAgent itself, which orchestrates the retrieval-generation process. The retrieval mechanism computes cosine similarity between the query embedding and all document embeddings, returning the top-k most semantically similar documents. This approach, while computationally efficient, treats all documents in a flat namespace and lacks domain-specific filtering or prioritization.
2. RAGDomainAgent - Domain-aware semantic retrieval: The RAGDomainAgent extends the baseline RAG approach by introducing explicit domain awareness through hierarchical document organization. This agent incorporates a DomainVectorStore that pre-partitions documents into distinct domains and subdomains. The vector store maintains separate embedding matrices per domain, enabling filtered similarity searches that retrieve documents exclusively from the identified domain. This filtering mechanism, while simple, significantly reduces irrelevant context injection by constraining the search space to semantically coherent document collections. The retrieval process includes an optional subdomain-level refinement, enabling granular filtering when metadata provides sufficient granularity. This domain-aware architecture demonstrates that explicit knowledge organization, even with simple embeddings, substantially reduces hallucination in specialized extraction tasks.
3. RAGDomainSysAgent - System-level context injection: The RAGDomainSysAgent introduces a fundamental architectural variation by relocating the retrieved glossary from the user prompt to the system prompt. This design decision reflects the hypothesis that contextual information provided at the system level receives different attention weighting from the language model compared to user-level context. The implementation maintains the domain-aware vector store from RAGDomainAgent but modifies the prompt construction strategy. Instead of augmenting the user message with glossary content, this agent injects the retrieved context directly into the system prompt through the glossary placeholder. The user message contains only the task instruction, while the system prompt provides the complete directive, including the professional role, extraction schema, source context, and retrieved domain knowledge.
4. RAGDomainGloveAgent - Optimized GloVe-based domain retrieval: The RAGDomainGloveAgent represents a significant optimization of the domain-aware approach, combining the lightweight efficiency of GloVe embeddings with the hierarchical domain organization introduced in RAGDomainAgent. This agent leverages the standard GloVe 6B.50 embeddings, which are predownloaded and cached locally. The agent introduces several key optimizations. First, the DomainGloveVectorStore precomputes embedding matrices for each domain as dense NumPy arrays (`embeddings_matrix`), enabling vectorized cosine similarity calculations instead of per-document loop computations. This batch-processing approach reduces retrieval latency by approximately 60% compared to the iterative similarity calculation in RAGDomainAgent. Second, the agent implements a document cap per domain (`max_docs_per_domain=5`) to maintain bounded memory consumption and search complexity. This configuration limits the embedding matrix size to a maximum of five documents per domain, ensuring that similarity searches remain computationally tractable even as the knowledge base expands. This implementation achieves the best balance between retrieval accuracy and computational efficiency among the GloVe-based variants, demonstrating that careful optimization of a simple embedding-based pipeline can achieve performance competitive with heavier dense retrieval approaches.
5. RAGDomainChunkingAgent - Asynchronous initialization with FAISS: The RAGDomainChunkingAgent represents the most technically complex implementation, incorporating

Table 1
Comparison of RAG pipeline implementations with model Llama 3.2 1B.

Feature	RAGAgent	RAGDomain-Agent	RAGDomain-SysAgent	RAGDomain-GloveAgent	RAGDomain-Chunking-Agent
Embedding	GloVe 50d	GloVe 50d	GloVe 50d	GloVe 50d	FastEmbed/BGE-small
Domain Awareness	No	Yes	Yes	Yes	Yes
Context Location	User prompt	User prompt	System prompt	User prompt	User prompt
Retrieval Index	Flat (in-memory)	Domain-separated	Domain-separated	Vectorized arrays	FAISS HNSW
Chunking Support	Whole documents	Whole documents	Whole documents	Whole documents	Tokens / Sentences / Paragraphs (best)
Async Loading	No	No	No	No	Yes
Fallback Search	No	No	No	No	Lexical matching
Retrieval Optimization	Sequential	Sequential	Sequential	Vectorized batch	Approximate Nearest Neighbors
Max Tokens per Document	Unlimited	500	500	300	500 (characters/tokens)
Document Cap (per domain)	Unlimited	Unlimited	Unlimited	5	3
Glossary Size (chars)	4000	4000	3000	4000	4000
Best F1-score	0.4916	0.5338	0.5209	0.5308	0.5358

three advanced features: asynchronous vector store initialization, FAISS-based efficient retrieval with HNSW indexing, and support for multiple chunking strategies. The asynchronous initialization pattern addresses a critical usability concern: the agent becomes immediately available for inference while the vector store loads in the background. Through a threading-based initialization mechanism protected by a lock, the agent can accept tasks and begin processing even if the embedding index is not fully loaded. The FAISS (Facebook AI Similarity Search) integration represents a significant performance optimization. The vector store builds an HNSW (Hierarchical Navigable Small World) graph index with configurable parameters (efConstruction=40, efSearch=16), enabling approximate nearest neighbor search with sub-linear time complexity. The index is persisted to disk and loaded from cache on subsequent runs, reducing initialization time from minutes to seconds. The caching mechanism also stores the document-store mapping, enabling fast retrieval without re-embedding. The chunking strategy consisted in pre-processes documents using three chunking methods (tokens, sentences, paragraphs). Each variant was tested separately, and the best-performing configuration was selected for evaluation. The F1 scores presented in Table 1 reflect the main characteristics of each RAG pipeline variant and the best performance achieved with the Llama 3.2 1B-Instruct model (via LM Studio).

`RAGAgent`, our RAG baseline implementation, demonstrates that even flat, non-domain-aware retrieval provides substantial improvements over zero-shot prompting. `RAGDomainAgent` adds domain-aware filtering, which yields a relative improvement of approximately 8.5% over the baseline `RAGAgent`, confirming the effectiveness of hierarchical knowledge organization.

`RAGDomainGloveAgent` achieved the highest F1 score among all RAG variants. This represents a 21.5% relative improvement over the baseline and validates the combination of GloVe-based retrieval

optimization with domain-aware filtering. Furthermore, repeated runs of this pipeline on the 1B model (F1 scores: 0.5298, 0.5266, 0.5359) demonstrate consistent performance.

`RAGDomainSysAgent` and `RAGDomainChunkingAgent` represent architectural explorations: system-level context injection and asynchronous FAISS-based retrieval with chunking support, respectively. While their performance did not achieve the best results, these variants provide valuable insights into alternative RAG architectures for future experimentation. The chunking approach consumes more RAM; consequently, depending on the model size and the constraints of local inference time, these models returned timeouts in the majority of cases.

3.4. Pipeline 3: Grammar-Constrained Decoding ("`grammar_cd`")

This pipeline uses the Outlines library [21] to constrain the LLM's output at the token generation level. Instead of post-processing or relying on the model to self-correct, GCD builds a context-free grammar from the target JSON schema and filters the model's logits at each step to only produce tokens that conform to the grammar.

A key task constraint is that each label-value pair in the output must be grounded in the context provided to the language model. Therefore, although using Outlines to filter the model's output improves overall quality, it is not sufficient on its own to fully address the challenge. In order to achieve better results, we proposed two strategies to ensure syntactically valid JSON output that combine Outlines with techniques to guarantee that the values remain strictly grounded in the input context, without hallucinations. The proposed techniques were as follows:

- Spanish instruction prompt: A Spanish prompt (the same one used in the Prompt-Eng pipeline) that instructs the language model to extract information exclusively from the context or text provided as input.
- English instruction prompt: The same prompt with semantically equivalent content in English.

Key aspects of our GCD implementation:

- The JSON schema provided in the task is compiled into an equivalent context-free grammar.
- Right-recursive grammar structures are prioritized to minimize computational cost, following the SAC principle [22].
- Generation uses a low temperature (0.0 to 0.2) to ensure deterministic, grammatical outputs.
- The system prompt is reduced to a minimal instruction, as formatting constraints are enforced by the grammar.

This method guarantees syntactically valid JSON, eliminating a primary source of extraction failures in SLM. The trade-off is a slight increase in per-token latency due to the grammar validation step.

4. Results and discussion

The three pipelines were evaluated on 40 extraction tasks across four domains. The experiments were conducted on a standard CPU (i5–10th generation, 16 GB RAM) via Ollama and LM Studio, as well as on T4 and L4 GPU with 16 GB and 28 GB RAM, respectively, via Ollama using Lightning.AI. To enable local execution, model quantization was set as follows: Q8_0 for Llama 3.2 1B; Q4_K_M for Llama 3.2 3B, Salamandra 7B, Qwen 2.5, Qwen3 14B, and Gema 4 E4B. The GCD pipeline was evaluated with Chain-of-Thought (CoT) reasoning enabled. The best results were achieved with a temperature of 0.0 across all experiments.

Table 2 reports the best micro-F1 scores obtained for each pipeline and model combination. Table 3 summarizes the best overall values per model. Table 4 reports the results of the best-performing variant of each pipeline on the test set.

Table 2
Best micro-F1 scores per pipeline and model on the starter dataset (40 tasks).

Pipeline	Model	Micro-F1
Prompt Engineering		
Prompt-Eng (XML)	Llama 3.2 1B	0.4075
Prompt-Eng (HFE)	Llama 3.2 1B	0.4229
Prompt-Eng (HFE)	Llama 3.2 3B	**0.5920** (reported by task organizers)
Retrieval-Augmented Generation		
RAG-Domain-GloVe	Llama 3.2 1B	0.5359
RAG-Domain-GloVe	Llama 3.2 3B	0.5975
RAG-Domain-GloVe	Llama 3.2 3B	0.6188 (reported by task organizers)
RAG-Domain-GloVe	Qwen2.5 14B	**0.7220**
Grammar-Constrained Decoding		
GCD (CoT, English)	Llama 3.2 1B	0.3498
GCD (CoT, Spanish	Llama 3.2 1B	0.5118
GCD (CoT, English)	Llama 3.2 3B	0.5053
GCD (CoT, Spanish)	Llama 3.2 3B	0.5293
GCD (CoT, Spanish)	Llama 3.2 3B	0.6308 (reported by task organizers)
GCD (CoT, English)	Salamandra 7B	0.4197
GCD (CoT, Spanish)	Salamandra 7B	0.4072
GCD (CoT, English)	Qwen2.5 14B	0.6451
GCD (CoT, Spanish)	Qwen2.5 14B	**0.6460**

Note: Prompt-Eng=Prompt Engineering; RAG = Retrieval-Augmented Generation; GCD = Grammar-Constrained Decoding; HFE = High-Fidelity Extraction; CoT = Chain-of-Thought. Bold values indicate the best performance achieved for each pipeline.

Table 3
Summary of best performance per model across all pipelines on the starter dataset.

Model	Best Pipeline	Micro-F1
Llama 3.2 1B	RAG-Domain-GloVe	0.5359
Llama 3.2 3B	GCD (CoT, Spanish)	0.6308
Salamandra 7B	GCD (CoT, English)	0.4197
Qwen2.5 14B	RAG-Domain-GloVe	0.7220

4.1. Analysis

The zero-shot Prompt-Eng pipeline is a robust and fast baseline. Even with the smallest Llama 3.2 1B model, this approach achieved a competitive F1 of 0.422. Its 100% completion rate on 40 tasks validates that a well-structured system prompt can reliably guide tiny models. The model retrieves more correct entities with the flat prompt of the high-fidelity approach than with the XML prompt for this model and dataset. The XML semantic zone variant showed particular strength in the medical domain, confirming the isolation hypothesis of [1].

RAG-Domain-Glove outperforms all other pipelines with the Qwen2.5 14B model, achieving an F1 of 0.7220 on the starter dataset. This method is computationally efficient (the GloVe index is small and fast) and directly tackles the vocabulary gap, leading to higher precision in specialized domains. This variant consistently outperforms the standard RAG and RAG-Domain variants. For Llama 3.2 1B, RAG-Domain-GloVe also achieves the best at 0.5359. This is notable given the lower parameter count.

Table 4
Test set (145 tasks) results and gap-closed metrics.

Model	Pipeline	Micro-F1	Gap Closed (%)
Gemma 4 E4B	Baseline	0.4061	—
	Prompt-Eng (HFE)	0.3513	0.0%
	RAG-Domain-GloVe	0.3750	0.0%
	GCD (CoT, Spanish)	**0.5559**	**25.22%**
Qwen 3 14B	Baseline	0.5604	—
	Prompt-Eng (HFE)	**0.6134**	**12.06%**
	RAG-Domain-GloVe	0.5652	1.09%
	GCD (CoT, Spanish)	0.5970	8.33%

Note: Gap Closed = $\max(0, (\text{F1}_{\text{system}} - \text{F1}_{\text{baseline}})/(1 - \text{F1}_{\text{baseline}}))$. This metric represents the proportion of the performance gap between the baseline and perfect performance (F1 $= 1.0$) that the system successfully closes.

The GloVe-based retrieval was exceptionally fast on CPU, proving that effective RAG for SLM does not require heavy dense retrievers; a well-curated domain glossary and a lightweight embedding model are sufficient. The addition of a glossary significantly reduced hallucinations in both technical and medical tasks. However, larger glossaries increase token usage and reduce CPU performance. When hardware constraints are not a concern, highly specialized glossaries enhance domain knowledge for specific fields.

The Grammar-Constrained Decoding pipeline provides a strong balance of performance, achieving its highest F1-score on the starter dataset of 0.646 with the Qwen2.5:14B model. The Llama 3.2 3B model achieves the best at 0.6308. This confirms that for complex, nested JSON schemas, enforcing structure at the token level is more reliable than relying on instruction-following alone, especially for SLM. For Llama 3.2 1B, standard GCD (0.5118) outperforms CoT variants (0.3498–0.3908). For Llama 3.2 3B, GCD (CoT, Spanish) at 0.5293 outperforms the English variant (0.5053). For Salamandra 7B, GCD (CoT, English) at 0.4197 slightly outperforms Spanish (0.4072). For Qwen2.5 14B, both variants achieve excellent performance (0.6451–0.6460), with Spanish slightly better.

The results of combining Spanish or English prompts with Outlines-filtered output reveal two key findings: (1) models with greater capacity (i.e., more parameters) yield better results, and (2) Spanish prompts closely aligned with the input context produce superior outcomes. The first finding is attributable to the stronger inference capabilities of larger language models, which help mitigate hallucinations. The second finding indicates that Spanish prompts perform better due to the alignment between the input text in Spanish (the context) and the system prompt proposed in this research.

Model size matters, but not linearly. The Salamandra 7B model underperformed compared to Llama 3.2 1B in the GCD pipeline. This suggests that raw parameter count is not the sole determinant; architectural choices and training data quality (especially for instruction-following in Spanish) play critical roles. Conversely, Qwen2.5:14B excelled, indicating that the 14B parameter scale may be a "sweet spot" for complex structured extraction when using GCD. Language also plays a crucial role. In this scenario, the JSON files contain Spanish instructions but English schema tags, which suggest bilingual models perform better. Since the model is tuned for Spanish text, the GCD pipelines with Spanish instructions score lower than those with English prompts, demonstrating that this model did not correctly process the English tags.

The Prompt-Eng (HFE) pipeline achieved the highest F1 scores for the Qwen 3 14B model on the test set, attaining the highest gap-closed value of 12.06%. This demonstrates the advantage of a larger model with greater knowledge and reasoning capabilities. For Qwen 3 14B, the GCD (CoT, Spanish) pipeline achieved an F1 of 0.60 with a gap closed of 8.33%, outperforming RAG-Domain-GloVe (F1 = 0.57, gap closed = 1.09%). These results suggest that for larger models, prompt-level constraints may be more beneficial than retrieval-based or decoding-based approaches. Notably, the baseline for Qwen 3 14B (0.56) was already strong, which explains the smaller relative gains. In contrast, the lower performance

observed with Gemma 4 E4B suggests that Qwen's multilingual capabilities make it better suited for Spanish-language prompts than Gemma.

The highest F1 score for Gemma 4 E4B was obtained by the GCD (CoT, Spanish) pipeline, with a gap-closed value of 25.22%. The other two pipelines did not improve upon the baseline, suggesting that this model responds negatively to prompt engineering and domain-based retrieval augmentation.

The worst results for the RAG pipeline were observed in the domains: `cultural_album_review`, `stem_observations`, `medical_trials`, `cultural_monuments`, `legal_case_timeline`, `cultural_theater`, `technical_security_incident`, and `environmental_assessments`. Additionally, some tasks timed out; the following domains failed to complete within the allocated time limit of 60 seconds: `cultural_album_review`, `stem_observations`, `legal_case_timeline`, `cultural_theater`, `technical_security_incident`, and `environmental_assessments`. The performance decline can be attributed to two main factors. First, the test domains include unseen domains with no corresponding knowledge in the retrieval index. Second, hardware limitations forced conservative token and character limits, reducing the information available in the vector store and loaded into the prompt.

5. Conclusions

This paper presented an evaluation of three model-agnostic techniques for enhancing small language models for structured information extraction. Our key findings are as follows:

1. Structured Prompt Engineering provides a strong, zero-shot baseline and excels with larger models. The Prompt-Eng (HFE) pipeline achieved competitive results (F1 = 0.42) with a 1B model on the starter dataset, with 100% task completion. Notably, it outperformed all other techniques and the baseline on the test set with Qwen 3 14B, confirming that instruction design is critical. This suggests that explicit structural constraints are more effective than domain-specific retrieval and grammar-constrained decoding when the base model already possesses strong reasoning capabilities.
2. Domain-aware RAG with lightweight embeddings is the most effective technique with a Llama 3.2 1B model (F1 = 0.54) on the starter dataset, significantly closing the gap with larger models. It also attained the highest F1 score (0.72) with Qwen 2.5 14B, demonstrating that targeted external knowledge is a powerful performance lever for SLM. This pipeline offers the optimal balance between performance and resource consumption, making it viable for most production environments. Moreover, for specialized fields such as medicine, law, and engineering, injecting domain-specific glossaries through RAG is a low-cost intervention that substantially reduces hallucinations.
3. Grammar-Constrained Decoding is highly efficient and robust. By constraining the output at the token generation level, GCD guarantees syntactic validity. It achieved the highest F1-score (0.65) with the Llama 3.2 3B model on the starter dataset and the best overall results with Gemma 4 E4B on the test set, confirming its robustness across different evaluation settings. Moreover, when latency is not the primary constraint, GCD with a model such as Qwen 2.5 14B offers state-of-the-art performance among open-weight models.
4. All three pipelines ran effectively on standard consumer hardware (CPU-only or basic GPU), proving that high-quality structured extraction is possible without costly cloud APIs and viable for production.
5. The results highlight that model-agnostic inference techniques, such as retrieval augmentation, grammar-constrained decoding, and structured prompting, can substantially bridge the performance gap between small and large language models on structured extraction tasks.

6. Recommendations

Based on our results, we offer the following recommendations:

- Integrating retrieval augmentation, grammar-constrained decoding, and structured prompting into a single pipeline is a promising direction for achieving the best overall performance.
- While GCD guarantees syntactic validity and RAG provides domain grounding, a hybrid approach that retrieves relevant schema constraints at decoding time could further improve accuracy on complex, nested JSON structures.
- The effectiveness of domain-specific glossaries depends on their quality and coverage. Future tooling should focus on semi-automated glossary generation from existing knowledge bases and documentation, reducing the manual effort required for domain adaptation.
- Given that prompt language had a measurable impact on performance, a systematic study of prompt language effects across typologically diverse languages could yield further gains and improve accessibility for non-English use cases.
- A systematic study comparing quantization methods and their effect on JSON validity and extraction accuracy would provide valuable guidance for developers deploying SLM in resource-constrained settings.

Acknowledgments

The authors would like to thank the organizers of the GenSIE 2026 task and IberLEF for providing the dataset and evaluation framework.

Declaration on Generative AI

During the preparation of this work, the author(s) used DeepSeek and Grammarly in order to: grammar and spelling checks, and code refactoring suggestions. After using these tools, the author(s) reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Elnashar, J. White, D. C. Schmidt, Prompt engineering for structured data: A comparative evaluation of styles and LLM performance, Artificial Intelligence and Autonomous Systems 2025 (2025). doi:10.55092/aias20250009.

[2] A. Piad-Morffis, et al., Overview of GenSIE at IberLEF 2026: Schema-Guided Information Extraction with Small Language Models in Spanish], in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2026), co-located with the 42nd Conference of the Spanish Society for Natural Language Processing (SEPLN 2026), CEUR-WS.org, 2026.

[3] A. Bonet-Jover, J. Á. González-Barba, L. Chiruzzo, Overview of IberLEF 2026: Natural Language Processing Challenges for Spanish and other Iberian Languages, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2026), co-located with the 42nd Conference of the Spanish Society for Natural Language Processing (SEPLN 2026), CEUR-WS.org, 2026.

[4] Z. Shen, D. Y.-B. Wang, S. S. Mishra, Z. Xu, Y. Teng, H. Ding, SLOT: Structuring the output of large language models, in: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, Association for Computational Linguistics, 2025, pp. 472–491. doi:10.18653/v1/2025.emnlp-industry.32.

[5] M. Mondal, L. Dolamic, P. Cudré-Mauroux, J. Audiffren, Two shots are enough: Reliable constrained generation with LLMs, in: Proceedings of the 2024 IEEE International Conference on Big Data, IEEE, 2024, pp. 4663–4672. doi:10.1109/BigData62323.2024.11402199.

[6] K. Mayilvaghanan, V. Nathan, A. Kumar, PROPEL: Prompt optimization with expert priors for small and medium-sized LLMs, in: Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing, Association for Computational Linguistics, 2025, pp. 272–302. doi:10.18653/v1/2025.knowledgenlp-1.25.

[7] NVIDIA, Llama Nemotron: Small language models for diverse tasks, NVIDIA AI, 2025. URL: https://build.nvidia.com/explore/discover, [Accessed: Jun. 12, 2026].
[8] Meta AI, Llama 3.2: Lightweight models for edge and mobile devices, Meta, 2024. URL: https://ai.meta.com/blog/llama-3-2/, [Accessed: Jun. 11, 2026].
[9] LangChain, Langchain: Build context-aware reasoning applications, LangChain, 2024. URL: https://www.langchain.com/, [Accessed: Jun. 10, 2026].
[10] A. Asai, S. Min, Z. Zhong, D. Chen, MetaGen: A unified framework for blended RAG generation, arXiv preprint arXiv:2405.15951 (2024).
[11] IBM, Granite: IBM's family of open-source large language models, IBM Research, 2024. URL: https://www.ibm.com/granite, [Accessed: Jun. 12, 2026].
[12] LangCache, LangCache Embed v3: Embedding models for semantic caching, LangCache, 2025. URL: https://langcache.ai, [Accessed: Jun. 12, 2026].
[13] Quentin Fuxa, PolyRAG: Agentic RAG platform purpose-built for small language models (slm). robust pdf/sql search, GitHub, 2025. URL: https://github.com/QuentinFuxa/PolyRAG.
[14] C. Chen, S. Borgeaud, G. Irving, J. B. Simon, FrugalRAG: Efficient retrieval-augmented generation with reinforcement learning, arXiv preprint arXiv:2410.13350 (2024). arXiv:2410.13350.
[15] Qwen Team, Qwen2.5: A party of foundation models, Alibaba Cloud, 2024. URL: https://qwenlm.github.io/blog/qwen2.5/, [Accessed: Jun. 12, 2026].
[16] N. Nundi, NuExtract: A foundation model for structured extraction, Numind, 2024. URL: https://about.nuextract.ai/blog/nuextract-a-foundation-model-for-structured-extraction.
[17] Liquid AI, LFM2: Liquid foundation models for structured extraction, Liquid AI, 2025. URL: https://www.liquid.ai/lfm2, [Accessed: Jun. 12, 2026].
[18] L. B. Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlícek, A. P. Lajarín, V. Srivastav, J. Lochner, C. Fahlgren, X.-S. Nguyen, C. Fourrier, B. Burtenshaw, H. Larcher, H. Zhao, C. Zakka, M. Morlon, C. Raffel, L. von Werra, T. Wolf, SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model, 2025. URL: https://arxiv.org/abs/2502.02737. arXiv:2502.02737.
[19] H. Prajapati, RAGZero-Core: Vectorless RAG for structured data, GitHub, ???? URL: https://github.com/HardikDev12/ragzero-core.
[20] C. Huyen, AI Engineering: Building Applications with Foundation Models, O'Reilly Media, 2024.
[21] Outlines, Outlines: Structured text generation, Online, 2025. URL: https://dottxt-ai.github.io/outlines/latest/, [Accessed: Jun. 12, 2026].
[22] F. Alpay, B. Senturk, Attention meets reachability: Structural equivalence and efficiency in grammar-constrained LLM decoding, arXiv preprint arXiv:2603.05540 (2026). URL: https://arxiv.org/abs/2603.05540. arXiv:2603.05540, submitted on 4 March 2026.
[23] Y. Zhang, Y. Li, Y. Liu, J. Li, X. Jia, Z. Li, G. Li, Lookahead-then-Verify: Reliable constrained decoding for diffusion LLMs under context-free grammars, Proceedings of the ACM on Programming Languages 1 (2026) 30. URL: https://arxiv.org/abs/2602.00612, also available at https://arxiv.org/pdf/2602.00612.
[24] J. Pennington, R. Socher, C. D. Manning, GloVe: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.