VerbaNexAI at GenSIE 2026: Intelligent Orchestration of Small Models for Zero-Shot Schema-Guided Information Extraction

Jesús Antonio Martínez Velandia[1,*], Jairo Enrique Serrano Castañeda[1], Edwin Alexander Puertas Del Castillo[1] and Juan Carlos Martínez Santos[1]

[1]GRITAS Research Group, Universidad Tecnológica de Bolívar, Cartagena de Indias, Colombia

Abstract

Schema-guided information extraction is one of the most demanding problems in modern natural language processing: given a text in natural language and an arbitrary JSON schema, the system must produce a structured object that is both semantically faithful to the source text and syntactically valid according to the schema. The complexity increases when operational constraints exclude supervised fine-tuning and limit the available models to compact variants with fewer than 14B parameters running without GPU acceleration. In this work, we present a system developed by the VerbaNexAI Lab team at Universidad Tecnológica de Bolívar (Colombia) for the GenSIE 2026 shared task in IberLEF 2026. The proposed architecture is centered on a *Smart Orchestrator* that manages four specialized extraction pipelines: Precision Master, based on expert prompt engineering; CoT Extractor, which implements Chain-of-Thought reasoning with self-correction; Ensemble RAG, which combines semantic retrieval with self-consistency; and Parallel Extractor, which decomposes cognitively dense schemas into subsets of fields processed in parallel. The orchestrator also integrates five robustness mechanisms: null-trap detection, strict schema conversion, semantic chunking, fuzzy enum matching, and a validation-guided repair loop. In the official *dry run* of 40 instances with the `llama-3.2-3b-instruct` model, our Parallel Extractor pipeline reached F1 = 0.6086, surpassing the official baseline of F1 = 0.6051. The Smart Orchestrator obtained F1 = 0.6062. Internal experiments on the development set (149 instances) also show that retrieval enrichment improves F1 by up to 30 % in medical and environmental texts, but can reduce F1 by up to 61 % in technical texts, evidence that motivates the orchestrator's *domain-aware* routing policy. These results show that adaptive orchestration at inference time can become a competitive advantage even when the base model capacity is fixed.

1. Introduction

Structured information extraction (*Information Extraction*, IE) has been a key part of natural language processing for decades [1]. Its goal is to identify the relevant entities, relations, and events in free text for a given application, and represent them in a computable way. However, classic approaches —based on patterns, grammars, or models with fixed ontologies— create a serious bottleneck: every new domain or target structure requires a costly cycle of design, annotation, and training [2].

The rise of large language models (LLMs) has opened the door to more general IE, without fixed ontologies or expensive supervised annotation stages. Recent studies show that LLMs can extract structured information in zero-shot or few-shot mode using carefully designed prompting [3, 4]. Even so, this paradigm has its own limits when operational constraints rule out access to large-scale models, require offline execution, impose strict limits on time and token budget, or demand that the output be valid JSON that follows an arbitrary schema seen for the first time at inference time.

The GenSIE 2026 shared task in IberLEF 2026 sets up exactly this scenario [5, 6]. Each instance consists of a Spanish text, a natural-language instruction, and a target JSON schema that may vary in complexity, domain, and lexical conventions. Participating systems must generate valid predictions using models with fewer than 14B parameters, without network access at evaluation time, with a

IberLEF 2026, September 2026, León, Spain

*Corresponding author.

jvelandia@utb.edu.co (J. A. M. Velandia); jserrano@utb.edu.co (J. E. S. Castañeda); epuertas@utb.edu.co (E. A. P. D. Castillo); jcmartinez@utb.edu.co (J. C. M. Santos)

0009-0001-5866-6986 (J. A. M. Velandia)

maximum context of 32K tokens, and a 60-second timeout per task.

These restrictions reflect a highly practical setting. In real applications, such as the digitization of clinical records, the analysis of legal contracts, the extraction of entities for information retrieval systems, or the input of automated agentic workflows, extraction systems must work with controlled resources, ensure data privacy without external dependencies, and adapt to changing schemas without retraining cycles [7]. GenSIE 2026 formalizes these requirements and makes it possible to evaluate solutions with comparative rigor.

The most naive approach to this scenario is to apply a single zero-shot prompting pipeline, standardized for all instance types. However, our central hypothesis is that different domains and schema configurations require qualitatively different inference strategies. Specifically:

- Medical and environmental texts benefit from retrieved examples, which provide recurring lexical patterns and help disambiguate specialized terminology.
- Technical and legal texts require extreme lexical precision: software names, legal articles, version identifiers, or enumeration values do not allow paraphrasing or analogical transfer from similar examples.
- Schemas with many fields exhaust the attention capacity of small models, which tend to omit fields or hallucinate invented values; in these cases, breaking the schema into smaller subsets improves coverage.
- Many schemas contain "tricky" fields: optional fields or fields with weak textual evidence where the model prefers to hallucinate rather than return `null`; these fields must be identified and handled proactively.

Based on this diagnosis, we designed in the VerbaNexAI lab a multi-pipeline extraction system centered on an Intelligent Orchestrator that applies the optimal strategy according to domain and schema complexity. The main contributions of this work are as follows:

1. We present a modular extraction architecture oriented to SLMs that competes without fine-tuning through four complementary pipelines coordinated by a domain-aware orchestrator.
2. We describe a set of robustness techniques for inference: null-trap detection, strict JSON schema conversion, semantic splitting, fuzzy matching of enumerations, and validation-guided repair.
3. We present empirical evidence from the official *dry run* showing that our Parallel Extractor outperforms the competition baseline.
4. We systematically analyze the effect of domain-based retrieval enrichment, showing why RAG benefits some domains and hurts others, and how the orchestrator takes advantage of this finding.

The rest of the article is organized as follows. Section 2 describes the GenSIE 2026 task and its formal constraints. Section 3 presents the architecture proposed by the VerbaNexAI team. Section 4 details the robust extraction techniques implemented by the system. Section 5 describes the experimental protocol. Section 6 reports the results. Section 7 analyzes the key findings. Section 8 closes the work with conclusions and future work.

2. Task Description

This section provides a formal characterization of the GenSIE 2026 shared task [5, 6]. We describe the problem formulation, the operational constraints imposed on participating systems, the domain diversity of the dataset, and the evaluation metric.

Table 1
Operational constraints of GenSIE 2026 and their design implications.

Constraint	Practical implication	Design decision
Small models (<14B parameters)	The system cannot rely on large-scale capacity to compensate for weak prompting or poor schema handling.	Use compact SLM-friendly pipelines with strong prompt engineering and robust post-processing.
Zero-shot, without fine-tuning	No task-specific supervised adaptation is allowed.	Design domain-aware prompting, retrieval-based augmentation, and schema normalization strategies that work without training.
Maximum context of 32K tokens	Long inputs and large schemas must fit within a bounded context window.	Apply semantic fragmentation, schema decomposition, and selective retrieval to reduce prompt length.
Time limit of 60 seconds per instance	Each inference decision must be fast and predictable.	Prefer lightweight routing, single-pass decoding when possible, and controlled parallelization.
Fully offline execution	No external API calls or online services are available at evaluation time.	Precache local embeddings, use local stores, and avoid dependencies on network access.

2.1. Problem Formulation

GenSIE 2026 (General-purpose Schema-guided Information Extraction) is a shared task in IberLEF 2026 focused on extracting information from Spanish texts under arbitrary and heterogeneous JSON schemas [5, 6]. Each instance is made up of three elements: (i) a natural-language instruction that describes the extraction goal, (ii) a source text that may vary in length, and (iii) a JSON schema that specifies the type, name, and constraints of each expected output field.

The system output must be a valid JSON object whose keys and values match the provided schema, whose textual values are faithful to the source text (without hallucinations), and whose missing fields are correctly represented (usually as `null` or empty arrays). Structural validity and factual faithfulness are both equally important for the evaluation metric.

2.2. Formal Constraints

Beyond the problem formulation, GenSIE 2026 imposes a set of strict operational constraints that directly shape every design decision in the system. These restrictions are not merely technical parameters: they collectively define a resource-constrained inference regime where trade-offs between accuracy, speed, and coverage must be carefully managed.

The operational constraints of GenSIE 2026 have a direct and deep impact on design decisions [5].

2.3. Domains and Diversity

The task instances cover at least eight domains (*medical, legal, technical, cultural, environmental, general, lifestyle, stem*), and in practice each one brings typical failures and different evidence criteria. This diversity is reflected not only in the data, but also in the system design itself: domain detection, which prioritizes `metadata.domain` and uses keywords as a fallback, together with domain-specific prompts, are central to maintaining quality under zero-shot and SLM constraints.

Variation across domains is not only thematic; it also changes the kind of "proof" expected by the evaluator and, therefore, the extraction strategy. Medical texts usually contain clinical terms, symptom lists, adverse effects, and descriptions that require high recall, so an analytical step or support from similar examples can help avoid missing entities. Legal texts, by contrast, often place evidence in long

clauses and normative formulas, and fields such as effective dates, jurisdiction, and parties tend to penalize paraphrasing strongly; for this reason, verbatim extraction and strict rules are preferred.

Technical texts frequently mention software names, licenses, versions, protocols, and fixed values, where the cost of inventing a version or checksum is high and conservative `null` policies together with exact matching in enums are more appropriate. Cultural texts introduce stylistic variation, including metaphors and opinions, as well as multiple entities, which makes it useful to control the tendency to summarize and to reinforce the selection of relevant text spans. Environmental and general news texts are dominated by facts, locations, and quantities such as victims or affected people, and the information is often spread across paragraphs, increasing the risk of omission.

Lifestyle texts, such as encyclopedic descriptions of dishes or practices, expose a common failure mode: the model tends to "complete" lists of ingredients, steps, or diet labels even when they are not explicitly present, which makes hallucination control critical in arrays and enums. Finally, STEM texts contain numbers, units, classifications, and scientific proper names, and the system must separate text-grounded knowledge from model knowledge to avoid external inference.

Diversity also appears in the schema type. Some domains favor arrays, such as lists of symptoms, entities, or ingredients; others require strict enums, such as categories, types, or frequencies; and others combine optional fields with a high incidence of null-traps, that is, fields that often do not appear in the text but that the model tries to fill in. In this project, this is addressed explicitly through prompt-level enum glossaries to enforce exact matching and reduce out-of-vocabulary values, null-trap detection to warn the model about fields prone to being invented, and post-processing to fill missing keys for anti-truncation and apply fuzzy matching to enums when typographical errors occur.

2.4. Metric

Evaluation is based on a schema-aware scoring scheme (*flattened schema scoring*), where each instance is compared against the *gold* at field level (and subfield level when nested structures are flattened), and the main metric reported is **Micro-F1**. In this setting, the metric is especially sensitive to: (i) structural or typing errors (for example, JSON Schema violations), (ii) mismatches in categorical values, and (iii) incorrect `null` assignments in optional fields (*null-traps*), since adding unsupported values or "filling in" missing information is heavily penalized.

As for the comparison criterion, **enumerated** and **boolean** fields are evaluated by **exact match**, while **arrays** and **free-text** fields allow a greater degree of tolerance, encouraging outputs that stay faithful to the source text, ideally *verbatim*, and consistent with the schema.

3. System Architecture

This section describes the architecture of the system developed by VerbaNexAI for GenSIE 2026. The design is based on the assumption that there is no single extraction strategy that is universally optimal for all domains and schema types. Therefore, the system adopts a modular, multi-channel architecture coordinated by a Smart Orchestrator that selects the most appropriate strategy at the time of inference, taking into account the task's resource constraints.

3.1. Overview

For this GenSIE challenge, the VerbaNexAI Lab proposal was developed in Python 3.13 as a modular framework for information extraction guided by *JSON Schema*. It provides an HTTP interface through FastAPI and a command-line interface for local evaluation and *benchmarking*. The system is distributed as a Docker image of about 758 MB, which precaches the `BAAI/bge-small-en-v1.5` embedding model during build time (via `fastembed`), thus meeting the offline execution requirement for the retrieval/embedding components.

At the agent level, the repository includes a baseline (`BasicAgent`) and three main pipelines aimed at the competition: Precision Master, CoT Extractor, and Ensemble RAG. It also includes a

Parallel Extractor (schema decomposition and parallel extraction) and a Smart Orchestrator, which dynamically selects the extraction strategy based on the domain and/or schema complexity.

3.2. Precision Master

Precision Master is the system's main zero-shot pipeline. It relies on expert prompt engineering through a *system prompt* with eight explicit rules (anti-hallucination, *verbatim* extraction, exact matching in rigid fields, conservative handling of `null`, among others), plus a glossary of enumerated values automatically derived from the schema to strengthen lexical compliance. To maximize compatibility with structured outputs, it uses `make_schema_strict` as a normalizer of the JSON Schema into a strict variant compatible with OpenAI-like backends. When the backend does not support strict structured output, the pipeline switches to a more permissive JSON output mode and relies on parsing and later validation of the generated JSON.

This pipeline has proven especially robust for rigid schemas and domains where lexical accuracy is critical (legal, technical, STEM).

3.3. CoT Extractor

CoT Extractor implements a two-stage flow. In the first call, the model produces a free-form analysis of the text, identifying relevant evidence and explicitly marking as `[NOT_FOUND]` those fields whose information does not appear literally in the source. In the second call, that analysis is reused as context to generate the final JSON according to the schema. A third self-correction step is activated only if the resulting output fails validation, incorporating the `jsonschema` error message to guide JSON repair.

This separation between (i) evidence localization and (ii) structuring reduces the cognitive load per call and tends to improve *recall* in long or dense texts; in the project's experiments, this pattern showed particular advantages in the medical domain.

3.4. Ensemble RAG

Ensemble RAG combines prompting with retrieval of similar examples from a local store (`ExampleStore`) built from the *starter set*. This store indexes instances using embeddings generated with `fastembed` and the `BAAI/bge-small-en-v1.5` model (with *fallback* to TF–IDF when `fastembed` is not available). For each new task, the two most similar examples are retrieved (optionally filtered by domain) and inserted as *few-shot* demonstrations in the prompt.

In its full variant, the pipeline applies *self-consistency* with multiple calls and aggregates outputs by field type: majority vote for enums and booleans, union for arrays, and selection of the candidate with the highest lexical overlap with the source text for free-text fields. However, due to time restrictions (60 s per instance), the system can adaptively degrade to a *single-shot* mode (N=1) when the latency or token budget is tight.

3.5. Parallel Extractor

Parallel Extractor addresses schemas with many properties through *schema decomposition*: it splits the *target schema* into field subsets ("chunks") and runs an independent extraction for each subset. The implementation uses a `ThreadPoolExecutor` to parallelize CPU calls (dynamically adjusting the number of *workers* based on `os.cpu_count()`). Partial results are merged by key union and then passed through a final post-processing step (filling missing fields and normalization) to produce a single JSON that conforms to the global schema.

This strategy reduces the cognitive load per call and is especially useful for dense schemas, where a single inference tends to omit fields or reduce faithfulness.

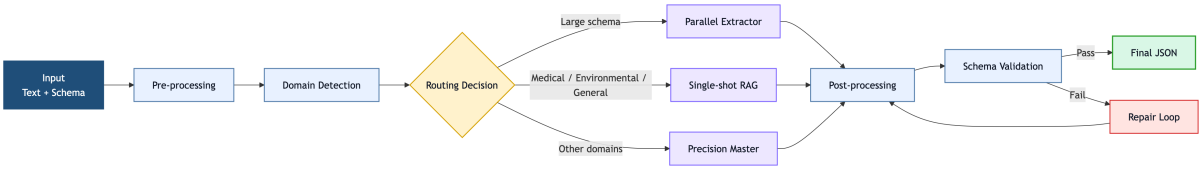

Figure 1: Decision flow of the SMART ORCHESTRATOR.

3.6. Smart Orchestrator

The SMART ORCHESTRATOR summarizes the system's routing logic under task constraints. Figure 1 shows how each instance is preprocessed, assigned to a domain, routed to the most suitable pipeline, and then refined through post-processing and validation-based repair before the final JSON is returned.

4. Robust Extraction Techniques

The pipelines described in Section 3 are complemented by a set of cross-cutting robustness techniques that address failure modes observed during development. These techniques operate at different stages of the extraction process — schema normalization, prompt construction, and output repair — and are applied selectively based on the characteristics of each instance. This section describes each technique in turn.

4.1. Parallel Extraction

Schema decomposition addresses the attention limit and cognitive overload in small models: by splitting the *target schema* into *chunks* with a few fields and solving each subset in independent calls, the probability of missing fields and the tendency to truncate or degrade the JSON structure are reduced. Pipeline-level parallelism (for example, using `ThreadPoolExecutor`) aims to amortize total latency cost, especially in CPU-based settings where several requests can overlap.

4.2. Null-Trap Detection

The `detect_null_traps` function targets a recurring failure mode in schema-guided extraction: models often prefer to invent plausible values for fields with weak or absent evidence rather than return `null`. This is particularly harmful in GenSIE 2026, where optional fields, nullable fields, and highly specific attributes such as hashes or exact dates can trigger hallucinated outputs that hurt both structural validity and Micro-F1. To mitigate this behavior, the system identifies these fields in advance and highlights them in the prompt with explicit warnings, reinforcing that `null` is a valid and often preferred output whenever the source text does not provide literal evidence.

4.3. Strict Schema Conversion

`make_schema_strict` transforms the input JSON Schema into a strict variant compatible with structured-output backends: it removes problematic constraints (for example, `pattern`), enforces `required` and `additionalProperties:false` in objects with `properties`, and recurses over nested structures and logical combinators (`anyOf`/`allOf`/`oneOf`). In addition, it handles `$ref` conservatively to meet structured-output format constraints. Overall, this normalization reduces validation failures and discrepancies due to subtle schema variations.

4.4. Domain Detection and Semantic Fragmentation

Domain detection prioritizes `metadata.domain` when available and, otherwise, applies a heuristic classifier based on keyword counts per domain. To control the context budget in long texts, a paragraph-based filter is applied using keywords derived from the schema (field names and relevant terms from their descriptions), keeping only fragments that are likely informative. This pre-processing reduces context without relying on external resources and lowers the risk of exceeding token limits.

4.5. Fuzzy Matching of Enumerations and Repair

In enumerated fields, fuzzy matching is applied against the set of allowed values, correcting typographical errors when similarity exceeds a threshold. Additional post-processing is then performed (for example, filling missing keys to mitigate truncation), and the resulting JSON is validated against the schema using `jsonschema`. If validation fails, a single guided self-correction pass is run, with a prompt that includes both the erroneous JSON and the validation error message.

5. Experimental Setup

This section describes the conditions under which the system was evaluated. We detail the execution environment and software stack, the language models used in internal and official evaluations, the datasets considered, and the metric implementation used for local benchmarking. All experiments were conducted under conditions that replicate the constraints of GenSIE 2026 as closely as possible.

5.1. Execution Environment

The system runs in a Docker image based on `python:3.13-slim` and precaches the `fastembed` embedding model (`BAAI/bge-small-en-v1.5`) during build time, which makes it possible to operate without downloads during evaluation. The stack includes FastAPI, an OpenAI-compatible client for local backends (for example, Ollama through an OpenAI-like endpoint), `jsonschema` for validation, and `typer` for the CLI and benchmarking interface. The design is aimed at CPU execution and at meeting the challenge's time and token-budget constraints.

5.2. Models

Internal evaluations on the development set were carried out mainly with `llama3.1:8b` served through Ollama via an OpenAI-compatible endpoint. In terms of decoding, the deterministic pipelines (for example, Precision Master and the *single-shot* RAG mode of the orchestrator) use temperature 0, while the *self-consistency* scheme of Ensemble RAG introduces sampling with temperature above zero to aggregate by voting and maximize robustness.

5.3. Datasets

Two main datasets are considered: the starter set (40 instances) used for the official *dry run*, and the full development set (149 instances) used for our internal experiments and to calibrate the orchestrator's routing rules.

5.4. Metric

Evaluation uses the official task metric based on *flattened schema scoring*, reported as **Micro-F1** (micro-averaged over fields and subfields after flattening the JSON according to the schema). The reference implementation is provided by the official challenge framework and is integrated into the system for local benchmarking. The results are reported both as aggregated F1 and broken down by domain, and they include the number of *strict-wins* per pipeline (instances where one pipeline gets the best score against the others on that task).

Table 2
Results of the official *dry run* (40 instances, `llama-3.2-3b-instruct`).

Pipeline	F1	Mean time (s/inst.)
Parallel Extractor	**0.6086**	0.87
Smart Orchestrator	0.6062	0.96
Precision Master	0.5942	0.50
Official baseline	0.6051	–

Table 3
Global results on the development set (149 instances, `llama-3.1-8b`).

Pipeline	Micro-F1	Mean TPS	Strict-wins
Precision Master	**0.0621**	0.469	45/149
Ensemble RAG	0.0618	0.468	40/149
CoT Extractor	0.0613	0.464	36/149
Ties between pipelines	–	–	28/149

6. Results

This section reports the performance of the system across two evaluation settings. The official dry run provides a direct comparison against the task baseline, while the internal experiments on the development set allow a more fine-grained analysis of each pipeline's strengths and weaknesses by domain.

6.1. Official Dry Run

Table 2 summarizes the results of the official *dry run* on the *starter set* with `llama-3.2-3b-instruct`.

Parallel Extractor exceeds the official baseline by approximately 0.0035 absolute F1 points, while the Smart Orchestrator also exceeds it, although by a smaller margin. Precision Master falls below the baseline in this subset, which suggests that the *starter set* schemas favor the decomposition strategy.

6.2. Development Set: Global Results

Table 3 presents the global results on the 149-instance development set with `llama-3.1-8b`.

The differences in aggregated F1 are small, which indicates that the capacity of the base model imposes a performance ceiling. Nevertheless, the distribution of *strict-wins* shows that no pipeline dominates universally.

6.3. Results by Domain

Table 4 breaks down the results by domain.

It can be seen that Precision Master dominates in the legal, technical, and STEM domains; Ensemble RAG dominates in cultural, environmental, and general; and CoT Extractor performs best in the medical domain. This justifies the design of the Smart Orchestrator.

7. Discussion

This section interprets the key findings from the experimental evaluation. We examine the domain-dependent effect of retrieval augmentation, analyze the role of orchestration when the base model capacity is the primary performance bottleneck, and discuss the specific challenges posed by the lifestyle domain. We close with a reflection on the limitations of the current approach.

Table 4
F1 by domain on the development set (149 instances, `llama-3.1-8b`).

Domain	P. Master	Ens. RAG	CoT Ext.	Winner
cultural	0.572	0.585	0.576	Ens. RAG
environmental	0.451	**0.484**	0.429	Ens. RAG
general	0.495	**0.502**	0.499	Ens. RAG
legal	**0.529**	0.511	0.505	P. Master
lifestyle	0.229	0.233	0.207	Ens. RAG †
medical	0.460	0.467	**0.492**	CoT Ext.
stem	**0.345**	0.323	0.339	P. Master
technical	**0.516**	0.492	0.464	P. Master

†Domain with the lowest global F1; the advantage is marginal.

7.1. Impact of RAG by Domain

The results show that RAG is beneficial in domains with recurring textual patterns and relatively stable terminology (medical, environmental, general), but harmful in contexts where lexical accuracy and local disambiguation are critical (technical, legal). This behavior is consistent with recent studies on the limits of parametric and non-parametric memory in LLMs [8, 9].

7.2. The Model Ceiling and the Role of Orchestration

The small global F1 difference between pipelines suggests that the SLM's capacity imposes a performance ceiling. Even so, variation by domain and by instance indicates that orchestration is still a source of improvement: an ideal orchestrator that always chose the best pipeline per instance would significantly outperform any individual pipeline.

7.3. The Lifestyle Domain as a Hard Case

The lifestyle domain shows the lowest F1 across all pipelines. We identified two main causes: the *starter set* contains very few instances from this domain, which limits the retrieval pool, and the texts have encyclopedic structures with long arrays, where models tend to hallucinate elements based on external knowledge.

7.4. Limitations

Our work has several limitations that should be acknowledged. First, the retrieval pool is both small and skewed toward a subset of domains, which constrains the quality and diversity of the examples available to the RAG-based pipeline. Second, the evaluation was carried out on a limited number of models, so the conclusions about portability across model families remain preliminary. Third, the routing policy is based on manual rules rather than learned decisions, which makes the system easier to interpret but also less adaptive to unseen patterns. Finally, the dataset may contain errors, including truncation issues and references to entities that are not visible in the source text, as reported by the organizers.

8. Conclusions and Future Work

We presented a schema-guided information extraction system in zero-shot mode, designed to operate under the strong constraints of GenSIE 2026 in IberLEF 2026. The architecture combines four specialized extraction pipelines under a Smart Orchestrator that adapts the strategy to the domain and to the schema complexity.

In the official *dry run*, our Parallel Extractor outperformed the official baseline, and the analysis on the development set showed that the usefulness of RAG is highly domain-dependent. It is beneficial

in medical and environmental texts, but it can be harmful in technical and legal texts. This result supports the view that structured extraction with SLMs should be treated as an orchestration problem rather than as the design of a single pipeline.

Future work will focus on expanding and balancing the retrieval pool with additional instances per domain, exploring other SLM families such as Qwen, Mistral, and Phi to assess the portability of the architecture, and investigating lightweight routing policies that respect time and resource constraints. We also plan to study the impact of quantization on the balance between performance and latency.

Acknowledgments

We thank the GenSIE 2026 organizing team for the benchmark design, the release of access to the *starter set*, and the execution of the official *dry run*. We also express special recognition to Karoll Pinto, systems engineer, for her constant support, technical feedback, and decisive help in defining the system architecture. Her judgment and perspective were an important pillar during the selection and refinement of the architectural decisions that shaped this work. This work was carried out within the activities of the GRITAS Research Group at Universidad Tecnológica de Bolívar, Cartagena de Indias, Colombia.

Declaration on Generative AI

During the preparation of this work, the `llama-3.2-3b-instruct` model was used as a component of the experimental system described. In addition, an AI-based assistant was used to support manuscript drafting and revision. All content was reviewed and validated by the authors, who take full responsibility for the accuracy, integrity, and interpretation of the results presented. See also the CEUR-WS policy on AI tools: https://ceur-ws.org/GenAI/Policy.html [10].

References

[1] J. Cowie, W. Lehnert, Information extraction, Communications of the ACM 39 (1996) 80–91. doi:10.1145/234173.234209.

[2] M. Mintz, S. Bills, R. Snow, D. Jurafsky, Distant supervision for relation extraction without labeled data, in: Proceedings of ACL-IJCNLP 2009, 2009, pp. 1003–1011.

[3] J. Wei, et al., Chain-of-thought prompting elicits reasoning in large language models, Advances in Neural Information Processing Systems 35 (2022).

[4] T. B. Brown, et al., Language models are few-shot learners, Advances in Neural Information Processing Systems 33 (2020) 1877–1901.

[5] GenSIE Organizers, GenSIE 2026: General-purpose schema-guided information extraction, https://uhgia.org/gensie/, 2026. IberLEF 2026 Shared Task. Accedido: junio de 2026.

[6] 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.

[7] NuMind, Nuextract: A structured extraction model, https://huggingface.co/numind/NuExtract, 2024. Accedido: junio de 2026.

[8] A. Mallen, et al., When not to trust language models: Investigating effectiveness of parametric and non-parametric memories, in: Proceedings of ACL 2023, 2023.

[9] E. Neeman, et al., Disentqa: Disentangling parametric and contextual knowledge with counterfactual question answering, in: Proceedings of ACL 2023, 2023.

[10] CEUR-WS Editorial Board, CEUR-WS policy on AI-assisting tools, https://ceur-ws.org/GenAI/Policy.html, 2025. En vigor desde el 1 de enero de 2025. Accedido: junio de 2026.