Inigo at IberLEF 2026 Task GenSIE: Schema-Guided Zero-Shot Information Extraction in Spanish with Small Language Models

Iñigo Goikoetxea Fanega[1,*], Mikel Sesma Sara[1] and Javier Lasheras Navas[2]

[1]Public University of Navarre (UPNA), Pamplona, Spain
[2]Research at Tracasa Instrumental, Pamplona, Spain

Abstract

We describe team **Inigo**'s submission to GenSIE 2026, the IberLEF shared task on general-purpose, schema-guided information extraction from Spanish text. The task asks a system to populate an arbitrary JSON schema, provided only at inference time, with information *grounded* in a source passage, returning `null` for absent fields. Because the task forbids fine-tuning and evaluates several small (<14B) open-weight models from different families, we focus entirely on inference-time engineering. We submit three model-agnostic pipelines that share a common backbone—grammar-constrained JSON decoding, a grounding-oriented system prompt, and a conservative rule-based post-processor that re-anchors values in the source—and differ in how much context they spend: `e226` (schema-typed few-shot with a per-field guide), `e231` (a compact zero-shot prompt tuned for the F1/token ratio), and `e232` (`e226` hardened for unseen schemas). On the test set, `e226` reaches a Micro-F1 of 0.74 with Qwen3-14B, while `e231` matches it closely using substantially fewer tokens. We report results across Qwen3-14B, Gemma-4-E4B, Llama-3.2-3B and Salamandra-7B to show the system is not tuned to a single model, and describe two schema-handling bugs we found, root-caused, and fixed during evaluation.

Keywords

Information Extraction, Zero-Shot Schema, JSON Schema, Small Language Models, Constrained Decoding, Few-Shot Prompting, Spanish NLP, IberLEF

1. Introduction

Modern agentic workflows depend on language models that can emit rigid, schema-valid structured data on demand. GenSIE 2026 (General-purpose Schema-guided Information Extraction) [1], a shared task at IberLEF 2026 [2], targets this capability in a deliberately hard setting: given a Spanish passage, a natural-language instruction, and a JSON Schema *seen only at inference time*, a system must produce a JSON object that strictly satisfies the schema, is faithfully grounded in the passage, and returns `null` for any field the text does not support. Roughly half of the private test schemas are entirely new, so memorising entity types is of no help; systems must read the field names, types, and descriptions of an unseen schema and act on them on the fly.

Two design choices of the task shape ours. First, the task enforces a strict *No-Training* policy: fine-tuning the model weights (LoRA, QLoRA, full fine-tuning) is prohibited, and only inference-time techniques—prompting, retrieval, constrained decoding, few-shot exemplars—are allowed. Second, each pipeline is evaluated against *several* small open-weight models (<14B) from different families, some of them held out and not disclosed before the results. The ranking rewards how much of the baseline-to-perfect gap a system closes, averaged over models, which discourages tuning to any single model.

This paper, therefore, deliberately departs from the (fine-tuning–centred) methodology of the broader thesis project this work stems from, and describes *only* the inference-time system submitted as team **Inigo**. Our contribution is a single model-agnostic agent exposing three pipelines that share a common,

IberLEF 2026, September 2026, León, Spain
*Corresponding author.
goicoechea.128710@e.unavarra.es (I. G. Fanega); mikel.sesma@unavarra.es (M. S. Sara); jlasheras@itracasa.es (J. L. Navas)
0009-0008-2068-6886 (I. G. Fanega); 0000-0002-2949-7909 (M. S. Sara); 0009-0007-4903-0103 (J. L. Navas)

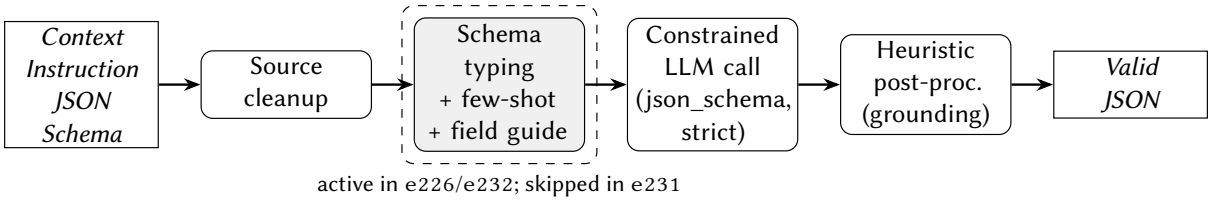

Figure 1: Shared inference backbone. All pipelines clean the source, call the model under grammar-constrained JSON decoding, and re-ground the result with a conservative post-processor. The dashed stage (schema-type inference, few-shot selection, per-field guide) is used by `e226` and `e232` and skipped by the compact zero-shot `e231`.

grounding-oriented backbone and trade context length for accuracy in different ways, together with a synthetic, schema-guided Spanish corpus—built and released as the few-shot retrieval pool—that supplies in-context exemplars for unseen schemas. Our best pipeline reaches a Micro-F1 of 0.74 with Qwen3-14B, and a compact zero-shot variant attains nearly the same accuracy at roughly $2.4\times$ fewer tokens.

2. Task and Evaluation

For each instance the system receives a *context* (a Spanish fragment from Wikipedia, news, scientific or legal sources), an *instruction* describing the extraction goal, and a *target schema* (a non-recursive subset of OpenAPI 3.0 / JSON Schema). The output must be a JSON object that (i) validates against the schema, (ii) copies information present in the context, and (iii) returns `null` for information that is absent—even when it is world-knowledge—so that parametric hallucinations are penalised [1].

Submissions are scored with *Flattened Schema Scoring*. Both gold and system outputs are flattened to dotted key–value pairs; values aligned by key are compared by type: rigid fields (numbers, dates, booleans, enums) require an exact match, free-text fields use a hybrid of multilingual sentence-embedding cosine similarity [3] and lexical overlap, lists are matched order-independently by greedy bipartite alignment, and any hallucinated value-vs-`null` mismatch scores zero. Per-model performance is a Micro-F1 over the test set. The primary leaderboard is not raw F1 but the average *gap closed* over the official baseline, $\max\big(0, (F1_s - F1_b)/(1 - F1_b)\big)$, averaged across evaluation models; a secondary leaderboard rewards the F1-to-token ratio. Soft budgets of 32K tokens and 60 s of user-code per instance (averaged over the test set) keep the comparison tractable. We report raw Micro-F1 and the F1/token ratio in this paper but do not compute our own gap-closed estimate; §6 explains why a locally reconstructed baseline is unsuitable for that purpose. The official gap-closed ranking will be computed by the organisers on their infrastructure with the authoritative baseline.

3. System

All three pipelines are exposed by a single model-agnostic agent that respects the `model` name passed by the evaluator and calls the organiser's OpenAI-compatible inference server with `temperature=0` and `stream=false`. They share the backbone in Figure 1 and differ only in the shaded context-construction stage.

Constrained decoding. Every call sets `response_format` to the target schema with `strict=true`, so the server enforces structurally valid JSON via grammar-constrained decoding [4]; the agent falls back to a non-strict call only if the server rejects the schema. This removes the "invalid JSON" failure mode and lets the pipelines spend their effort on content rather than syntax.

Grounding-oriented prompt. The system prompt instructs the model to extract *only* from the text, never to use external knowledge, to return `null` for unsupported fields, to copy values verbatim

(preserving capitalisation and date formats), to use the exact enum strings of the schema, and to be exhaustive over array fields. The explicit "prefer `null` over invention" rule directly targets the task's hallucination penalty.

Conservative post-processing. The raw model object is passed through a layered, schema-aware rule-based fixer. It canonicalises null aliases ("N/A", "desconocido", empty strings) to `null`; coerces and normalises rigid types; fuzzy-matches enum strings to the schema's allowed values; and re-anchors free-text and verbatim values in the source passage using a sliding-window RapidFuzz token-set match, coercing a nullable value to `null` (or dropping an ungrounded list item) when no supporting span is found. Fields whose schema description marks them as inferred, computed, or summarised are exempt from the grounding check. The fixer is conservative by construction: when no high-confidence correction applies, it returns the original value, so it can only add grounding signal, not remove correct content.

3.1. Synthetic corpus for few-shot retrieval

Because the test schemas are unseen, useful in-context exemplars cannot be drawn from a fixed ontology. The task explicitly permits synthetic data for few-shotting, provided the methodology is reported and the data released. We therefore build a synthetic, schema-guided Spanish corpus that serves *only* as a few-shot retrieval pool—no model weights are ever updated. The corpus is produced by a five-stage pipeline adapted from GuideX [5] to the GenSIE record format. (S0) Heterogeneous Spanish passages (encyclopaedic, journalistic, scientific, legal, medical, cultural) are extracted from Wikipedia [6], kept as self-contained text, and tagged with a domain. (S1) A teacher LLM (Llama-3.1-70B-Instruct) produces a free-form structured summary, key facts, candidate enumerations and—critically—*null-trap candidates*: questions the passage cannot answer. (S2) For each passage it synthesises a *unique* JSON Schema and natural-language instruction tuned to that passage; this per-passage uniqueness is the key departure from GuideX, since it forces a downstream system to read schema cues rather than memorise a fixed ontology. (S3) A further low-temperature call extracts a schema-valid JSON instance, retrying with validator feedback on failure. (S4–S5) Non-LLM quality control rejects malformed, degenerate (excess-null) or under-specified records and packages survivors as GenSIE instances.

The resulting pool contains ≈3,980 schema–instance pairs over ≈3,275 distinct schemas (each reused ≈1.2 times) across eight domains, with high structural coverage: 99.9% carry a nullable field with a null-trap, 91.5% contain arrays, 89.1% nested objects and 88.7% enums, and declared complexity spans L1–L10 (peaking at L4–L7). At inference, `e226`'s selector retrieves $k{=}3$ exemplars of the inferred schema type from this pool; `e232` instead retrieves from the smaller organiser-authored pools, and `e231` uses no exemplars. The corpus is released with the code, as required for augmented data.

3.2. The three pipelines

These three are the survivors of a broader development-time search over inference-time technique families (Section 5.1); each occupies a distinct point on the accuracy/cost/robustness trade-off.

`e226` — schema-typed few-shot. Builds context by (1) inferring a coarse schema "type" from field-name fingerprints, (2) selecting $k{=}3$ in-context exemplars of that type from the synthetic retrieval pool of Section 3.1 with a diversity heuristic (one long-array, one null-free, one null-bearing example, excluding the current instance to avoid leakage) [7, 8], (3) appending a per-field *field guide* that re-surfaces each field's description, enum options and nullability, and (4) prepending a small domain-specific addendum. This is our accuracy-oriented pipeline.

`e231` — compact zero-shot. Drops the entire few-shot/field-guide stage in favour of a six-line system prompt encoding the same grounding rules, keeping only source cleanup and the post-processor. It

is explicitly optimised for the F1/token secondary leaderboard: it spends roughly one call's worth of tokens and still benefits from constrained decoding and re-grounding.

e232 — hardened for unseen schemas. A variant of `e226` designed for the test set's ≈50/50 split of seen/unseen schema types. It draws exemplars from the organiser-authored development and starter pools, replaces the lenient schema-type matcher with a *strict* one (requiring a Jaccard overlap ≥ 0.5 and ≥ 4 shared fields) and gates the domain addendum on a confident match; when no type is matched (an unseen schema) it falls back to a hybrid structural + TF-IDF exemplar selector and keeps the neutral base prompt, so it never pushes an unseen schema toward the wrong type.

4. Experimental Setup

We evaluate with the official `gensie eval` harness over the released test set (N=145 instances) under Flattened Schema Scoring. We report two views. As the organisers require, our *primary test* results use the two mandated models, Gemma-4-E4B and Qwen3-14B. To verify that the system is genuinely model-agnostic and not tuned to one family, we also run all three pipelines across the published/recommended models of three different families and compute scales: Qwen3-14B (medium), Salamandra-7B (a Spanish-native model) [9], and Llama-3.2-3B (tiny) [10, 11]. For local development we use the lightweight `bge-small` embedder; the official leaderboard uses a heavier multilingual model, so our absolute free-text scores are a lower bound on the official ones.

Serving stack and reproducibility. The submission is a single Dockerised service (Python 3.13, FastAPI) exposing the competition's OpenAI-compatible `/run` and `/info` endpoints; per the hardware rules the agent logic runs CPU-only, and all GPU work is confined to the inference server it calls. The results in this paper were produced by serving each model with vLLM [12]—whose paged-attention KV-cache and continuous batching let us run the evaluation concurrently—under xgrammar grammar-constrained decoding, as SLURM-scheduled Singularity containers on the BSC MareNostrum 5 supercomputer, one NVIDIA H100 GPU per model (tensor-parallel size 1; every evaluated model fits on a single GPU), reached from the agent host over an SSH tunnel that mirrors the challenge's isolated single-endpoint design. The Dockerfile, agent, serving scripts and synthetic corpus are released (Section 7).

5. Results

5.1. Development-set exploration

The three submitted pipelines are the endpoint of a systematic search, run during development on the starter set (40 instances) across four families of inference-time technique: an off-the-shelf structured-outputs baseline, chain-of-thought prompting, schema-aware multi-step reasoning, and few-shot retrieval over the synthetic pool of Section 3.1. Table 1 reports the best Micro-F1 reached within each family (over 13 variants in total; full per-variant results are in the companion thesis). Few-shot retrieval is strongest on *every* model and schema-aware reasoning is a consistent second, while plain chain-of-thought roughly doubled compute for no F1 gain and was dropped. The submitted pipelines combine the two winning ingredients: `e226` and `e232` pair few-shot retrieval with the schema-aware post-processor, and `e231` keeps the cheap, baseline-like path.

5.2. Test-set results

Table 2 reports precision, recall and Micro-F1 for all three pipelines on all four evaluated models, with the final pipeline code (the two schema-handling fixes of Section 6); Gemma-4-E4B and Qwen3-14B are the two organiser-mandated models, and Llama-3.2-3B and Salamandra-7B are published-recommended models we add to probe cross-family generalisation. Figure 2 shows the same F1 values at a glance.

Table 1
Development-time exploration on the starter set (40 instances): best Micro-F1 within each inference-time technique family, per model (13 variants tested in total; best per column in bold). This is a design-selection experiment, not a test-set result.

Technique family	Qwen3-14B	Llama-3.2-3B	Salamandra-7B
Baseline (structured outputs)	0.742	0.685	0.470
Chain-of-thought	0.758	0.670	0.515
Schema-aware reasoning	0.765	0.685	0.540
Few-shot retrieval	**0.785**	**0.731**	**0.549**

Table 2
Test-set precision, recall and Micro-F1 for all three pipelines across all four evaluation models, with the final pipeline code. Gemma-4-E4B and Qwen3-14B are the organiser-mandated models; the best pipeline per model (Micro-F1) is in bold.

Model	Pipeline	P	R	Micro-F1
Qwen3-14B	e226	0.760	0.714	**0.736**
	e231	0.744	0.695	0.719
	e232	0.765	0.660	0.709
Gemma-4-E4B	e226	0.770	0.770	0.770
	e231	0.771	0.759	0.765
	e232	0.777	0.776	**0.776**
Llama-3.2-3B	e226	0.657	0.632	0.644
	e231	0.672	0.640	**0.655**
	e232	0.686	0.600	0.640
Salamandra-7B	e226	0.000	0.000	0.000
	e231	0.500	0.453	**0.475**
	e232	0.624	0.194	0.295

Table 3
Efficiency on Qwen3-14B: average tokens per instance and accuracy per 1K tokens (higher is better).

Pipeline	Micro-F1	Avg. tokens/inst.	F1 / 1K tok.
e226	0.736	10,095	0.073
e231	0.719	4,188	**0.172**
e232	0.709	10,104	0.070

Three patterns stand out. (i) The *best* pipeline is model-dependent: few-shot e226 wins on Qwen3-14B, the hardened e232 on Gemma-4-E4B, and the compact e231 on both smaller models—no single recipe dominates, which is exactly what the task's multi-model ranking rewards. (ii) The much smaller Gemma-4-E4B matches or beats the 14B Qwen3 on all three pipelines (0.765–0.776 vs. 0.709–0.736), evidence that the inference-time scaffolding, not raw model scale, drives most of the accuracy. (iii) Salamandra-7B is unusable under the few-shot pipelines (e226 = 0.000, e232 = 0.295) but strong under e231 (0.475); this is a hard 8K-token context limit, not a quality gap, analysed in Section 6. The high-precision, low-recall signature of Salamandra e232 (P=0.62, R=0.19) is its fingerprint: the instances that fit the window score well, the rest hard-fail.

Efficiency. Table 3 reports average tokens per instance and the F1-per-1k-tokens ratio for Qwen3-14B. The compact e231 uses well under half the tokens of the few-shot pipelines (≈4.2K vs. ≈10.1K) for a Micro-F1 within two points of e226, so its F1/token ratio is roughly 2.4× higher. All pipelines stay well inside the 32K-token soft budget.

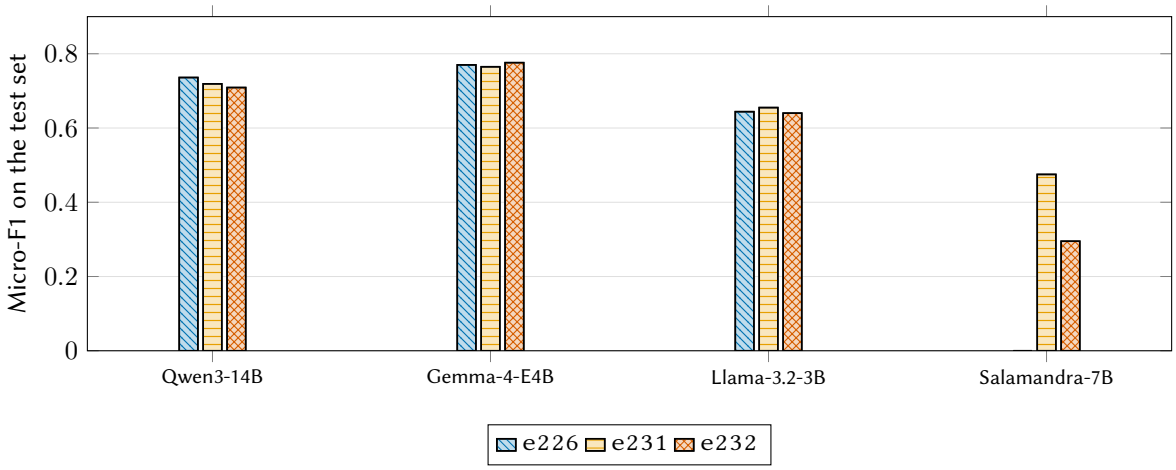

Figure 2: Micro-F1 of all three pipelines across all four evaluated models, with the final pipeline code. Salamandra-7B's near-zero `e226`/`e232` bars reflect its 8K-token context window overflowing on this test set's long passages (§6), not a quality gap.

6. Discussion

Accuracy vs. cost. The few-shot pipelines buy a couple of F1 points on the strongest models but at 2.4× the tokens (Table 3) and, more importantly, are *fragile* on weaker ones—collapsing outright on Salamandra. For deployments that prize the F1/token ratio, or that must run on small models, the compact `e231` is the safer default.

Distribution shift on the test set. An analysis of the released test partition versus the development data shows a clear shift: test passages are about three times longer (median ≈549 vs ≈247 words), expected outputs are denser, a new `research` domain appears while `lifestyle` disappears, and declared complexity moves toward the high end. This favours pipelines that read long contexts and remain structurally stable, and partly explains why exemplar-heavy prompting does not dominate: on long inputs the extra exemplars compete for the model's attention budget without adding schema-specific signal for unseen types.

Where the system wins and loses. Table 4 breaks Micro-F1 down by domain for each mandated model's best pipeline. Three things stand out. First, `research`—the one domain *absent* from development, i.e. fully out-of-distribution—is the hardest on both models (≈0.60), a direct measurement of the zero-shot-generalisation difficulty at the core of the task. Second, the large `stem` gap (Qwen 0.656 vs. Gemma 0.826) is the recursive-schema finding of Section 6 surfaced per domain: `stem` contains the ten recursive biology-taxonomy schemas, six of which time out under Qwen's few-shot pipeline (dragging the domain down) but resolve cleanly on Gemma. Third, `cultural` and `technical` are the easiest domains on both models, while the small `environmental` domain (n=7) is the weakest after `research`.

Context-window overflow on Salamandra-7B. The same length shift has a sharper, structural consequence for Salamandra-7B, whose server enforces an 8,192-token context window. On this test set, `e226`'s few-shot exemplars, field guide and (now longer) source passage push the prompt past that limit on effectively every instance, and the inference server rejects the request outright (`maximum context length is 8192 tokens`, requested up to ∼16K); with no fallback to a shorter prompt on this error, every such call becomes a hard failure, and `e226` scores $F1$=0.000 on Salamandra. `e232` overflows less often (60% of instances, vs. effectively all for `e226`) because its stricter schema-type gating sometimes skips the domain addendum, and still reaches 0.295. This is not a new failure mode: the companion thesis reports the same 8K window overflowing the richest zero-shot pipeline on 14 of

Table 4
Per-domain Micro-F1 on the test set for each mandated model's best pipeline (Qwen3-14B `e226`, Gemma-4-E4B `e232`); n is the number of instances per domain. Rows sorted by Gemma F1.

Domain	n	Qwen3-14B	Gemma-4-E4B
`cultural`	38	0.821	0.824
`technical`	12	0.801	0.824
`stem`	25	0.656	0.826
`general`	7	0.748	0.780
`legal`	22	0.731	0.752
`medical`	24	0.744	0.751
`environmental`	7	0.552	0.623
`research`	10	0.612	0.604

149 development-set tasks and works around it by selecting a lighter, non-overflowing prompt for that model. The test set's $\approx 3\times$ longer passages (previous paragraph) turn an occasional overflow into a near-total one for the few-shot pipelines, and are exactly why the compact `e231`—which never exceeds a few thousand tokens—is the only pipeline that remains usable on Salamandra (0.475).

Robustness. `e232`'s strict matcher and gated addenda are insurance for the unseen half of the test set: they prevent the pipeline from confidently mistyping a novel schema, at the cost of slightly lower F1 on the seen half where `e226`'s lenient matching helps. The conservative post-processor is the other robustness lever—because it only edits values it can re-ground in the source, it improves grounding without risking correct content.[1]

Grounding over-strictness on paraphrased fields. The test set's denser, more nested outputs ("Distribution shift" above) stress-test the post-processor's grounding check in a way the development set rarely did. Fields such as a field-worker's `notes`—natural-language content that is legitimately a light paraphrase of the source, not a literal quote—sit inside deeply nested object arrays (e.g. one `species_observed` entry per animal). On a sample of instances where the official zero-shot baseline outscored our pipelines despite extracting the same number of fields, we found a genuine bug: the post-processor's string-grounding check (`_string_grounded`) required an *exact* substring match (after case/diacritic folding) and had no fuzzy fallback, unlike every other layer of the post-processor. Concretely, for one instance the source stated *"cantos territoriales de Bubo bubo (búho real) procedentes del talud calcáreo situado ∼400 m al norte"*; the model correctly summarised this as *"cantos territoriales desde talud calcáreo al norte"*, and the exact-match check flagged it as ungrounded and coerced it to `null`—on this instance, repeatedly, across multiple sibling items in the same array. We fixed this by giving `_string_grounded` the same fuzzy span-matching fallback (RapidFuzz `token_set_ratio`, threshold 85) already used elsewhere in the post-processor, and confirmed at the unit level that it restores this specific class of value while still rejecting a genuinely fabricated string. However, a controlled test set-wide comparison—replaying the *same* captured raw model output through the pre-fix and post-fix post-processor for all 145 instances, which isolates the fix from Gemma's run-to-run generation variance (temperature 0 does not guarantee bit-identical output across separate inference sessions)—shows the bug fires rarely enough on this test set to leave the aggregate score unchanged (ΔF1 $= 0.0000$; only 3/145 instances produce a different output at all, and none of the three differences move the score). We keep the fix, since it removes a real and unambiguous source of unwarranted null-coercion and cost nothing to apply, but it does not explain why the baseline outscores our pipelines on Gemma; a second, distinct and *unfixed* issue—the post-processor occasionally *expanding* an already gold-exact short answer into a longer, lower-scoring sentence via its span-recovery logic—is a more likely contributor, left as a limitation for future work given the time available for this submission.

[1] During the formal evaluation our container failed the organisers' `/info` healthcheck (a start-up dependency-resolution issue); per the organisers' request we re-ran the pipelines on the mandated models and submit the annotated outputs directly.

Recursive schemas. One test family—biology taxonomy trees, where a node `$ref`s itself through a `children` list—is *recursive*, even though the task specification defines schemas as "a strict, non-recursive subset of the OpenAPI 3.0 specification… [to ensure] compatibility with standard grammar-constrained decoding techniques" [1]. Our schema compiler inlined `$ref` pointers with no cycle guard, so building the constrained-decoding grammar for these schemas raised a recursion error; the call then silently fell back to an unconstrained one whose free-form output did not parse, and all ten instances scored zero on every model—about 7% of the test set. We fixed this with a depth-bounded ref-resolution pass that inlines each definition at most five levels deep and caps deeper recursion with a permissive node, restoring constrained decoding on recursive schemas; this raised Micro-F1 on every pipeline/model combination we re-ran (e.g. `e226`: $+0.008$ on Qwen3-14B, $+0.029$ on Gemma-4-E4B). A residual issue remains on Qwen3-14B: with the depth-5 cap, the few-shot pipelines (`e226`, `e232`, longer prompts) time out (∼300 s, the harness limit) on several of the ten biology instances, while the compact `e231` and Gemma-4-E4B resolve all ten cleanly—compiling and running a constrained-decoding grammar for a recursive schema scales with both grammar depth and prompt length, and the cost depends on the inference backend. We did not reduce the depth cap further to chase this residual case, and report it as a known limitation: this specific schema violates the task's own non-recursion guarantee, and the point where our system still struggles is exactly the point that guarantee exists to prevent.

Local baseline reconstruction. To estimate the primary gap-closed leaderboard locally, we reconstructed the starter kit's reference `BasicAgent` (not the organisers' actual implementation, which is not distributed) as a fourth, separate pipeline. This reconstruction sends the raw `target_schema` unmodified—bypassing the schema-cleaning step described under "Constrained decoding" in Section 3—and has no exception handling around the model call. Against the Qwen3-14B backend (vLLM with xgrammar), this fails on 87 of 145 instances (60%, spread across nearly every domain) with `Error code: 400 - "The provided JSON schema contains features not supported by xgrammar"`; since the reconstruction does not catch this, each failure becomes an unhandled server error and the instance scores zero, deflating the local baseline to F1$= 0.36$ and inflating a naively-computed gap-closed to an unrepresentative $\approx 59\%$. Against the Gemma-4-E4B backend, the identical reconstruction never fails (0/145) and scores F1$= 0.805$, *above* all three of our own pipelines, which would put a naively-computed gap-closed at 0%. Neither number reflects system quality: the contrast measures whether a given backend's constrained-decoding engine accepts an unmodified JSON Schema, which is precisely why our own three pipelines clean the schema before every call. We therefore do not report a self-computed gap-closed figure; the official ranking will use the organisers' authoritative baseline on their own infrastructure.

An external, imperfect anchor. Lacking a trustworthy self-computed baseline, the closest thing we have to independent evidence that the engineering mattered is the organisers' own published reference: on the 40-instance starter set, their zero-shot baseline scores F1$= 0.295$ for Llama-3.2-3B-Instruct and F1$= 0.455$ for the far larger Llama-4-Maverick [1]. This is not a same-set comparison—the starter set is smaller and, by our own distribution analysis, easier than the test set (shorter passages, shallower nesting)—so it cannot substitute for an official test-set gap-closed. But for the one model shared between both references, Llama-3.2-3B, the direction is informative: our pipelines reach F1$= 0.64$–0.66 on the *harder* test set, versus 0.295 for the organisers' own zero-shot baseline on the *easier* starter set. If the harder set works against us, as our distribution analysis suggests it should, this comparison likely *understates* the true gap closed rather than overstating it. We report it as suggestive context, not as a substitute for the official ranking.

7. Conclusion

We presented team Inigo's inference-time approach to GenSIE 2026: a single model-agnostic agent with three pipelines over a shared backbone of constrained JSON decoding, grounding-oriented prompting,

and conservative re-grounding. The few-shot `e226` pipeline reaches 0.74 Micro-F1 with Qwen3-14B, while the compact zero-shot `e231` attains comparable accuracy at a fraction of the token cost and generalises better to smaller models. Results across three model families confirm that the system is not tuned to a single model, and that no single prompting strategy dominates across the compute tiers the task spans—a finding that argues for submitting complementary pipelines rather than one; on Salamandra-7B, whose 8K context window the few-shot pipelines routinely overflow on this test set, `e231` is not merely the better choice but the only usable one. Several discussion points above are debugging findings as much as design findings—a recursive-schema compiler crash, an overly strict grounding check, and a hard context-window limit on the smallest model—each traced to a concrete root cause. Their measured impact differed sharply: the schema-recursion fix recovered an entire test family that previously scored zero on every model, the context-window limit collapses `e226` to zero on one model specifically, while a controlled, test set-wide measurement showed the grounding fix, though a genuine bug, changes aggregate F1 negligibly on this test set. We report all three regardless, because knowing a fix does *not* move the metric is as useful as knowing one does, and because a single unhandled schema shape or resource limit can dominate an aggregate score as much as any prompting choice.

Acknowledgments

We thank the GenSIE 2026 organisers for the task, data, and evaluation tools. This work was carried out at the Public University of Navarre (UPNA).

Online Resources

The full system—agent, Dockerfile, SLURM/serving scripts and the synthetic few-shot corpus—is released under the MIT license at the team's repository, made public after the official results are announced.

Declaration on Generative AI

During the preparation of this work, the authors used generative AI assistants (ChatGPT, Claude) in order to perform grammar and spelling checks and to improve the readability of the text. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] 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.

[2] 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.

[3] N. Reimers, I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 3982–3992. doi:10.18653/v1/D19-1410.

[4] B. T. Willard, R. Louf, Efficient Guided Generation for Large Language Models, 2023. doi:10.48550/arXiv.2307.09702.

[5] N. D. L. Fuente, O. Sainz, I. García-Ferrero, E. Agirre, GuideX: Guided Synthetic Data Generation for Zero-Shot Information Extraction, 2025. doi:10.48550/arXiv.2506.00649.

[6] Wikimedia Foundation, Wikimedia downloads, 2023. URL: https://dumps.wikimedia.org, spanish Wikipedia dump dated 2023-11-01 (`20231101.es`), pre-processed and distributed via Hugging Face Datasets (`wikimedia/wikipedia`).

[7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, 2020. doi:10.48550/arXiv.2005.14165.

[8] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, W. Chen, What Makes Good In-Context Examples for GPT-3?, 2021. doi:10.48550/arXiv.2101.06804.

[9] A. Gonzalez-Agirre, M. Pàmies, J. Llop, I. Baucells, S. D. Dalt, D. Tamayo, J. J. Saiz, F. Espuña, J. Prats, J. Aula-Blasco, M. Mina, I. Pikabea, A. Rubio, A. Shvets, A. Sallés, I. Lacunza, J. Palomar, J. Falcão, L. Tormo, L. Vasquez-Reina, M. Marimon, O. Pareras, V. Ruiz-Fernández, M. Villegas, Salamandra Technical Report, 2025. doi:10.48550/arXiv.2502.08489.

[10] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, LLaMA: Open and Efficient Foundation Language Models, 2023. doi:10.48550/arXiv.2302.13971.

[11] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, T. Zhu, Qwen Technical Report, 2023. doi:10.48550/arXiv.2309.16609.

[12] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, I. Stoica, Efficient Memory Management for Large Language Model Serving with PagedAttention, in: Proceedings of the 29th Symposium on Operating Systems Principles, SOSP '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 611–626. doi:10.1145/3600006.3613165.