Few-Shot Prompting with Schema-Aware Post-Processing for Zero-Shot Structured Information Extraction in Spanish

Krishan Heredia[1]

[1]Eötvös Loránd University, Hungary

Abstract

I present my submission to the GenSIE 2026 shared task on General-purpose Schema-guided Information Extraction. My system addresses the challenge of extracting structured JSON from Spanish texts using small language models (sub-14B parameters) with zero-shot schemas provided at inference time. The proposed pipeline, `schema_dynamic`, uses English-language few-shot prompting with schema-agnostic post-processing that includes validation-guided self-correction, entity reordering by label priority, type normalization, and hallucination filtering. A distinguishing feature is dynamic label derivation: allowed labels are read directly from the schema enum at runtime rather than hardcoded, allowing the pipeline to adapt to schemas not seen during development. I evaluate on the official GenSIE test set and report the published results. My pipeline achieved F1 scores of 0.8142 and 0.8076 on Gemma 4 E4B and Qwen3-14B respectively, closing 12.0% of the baseline-to-perfect gap. These results demonstrate that carefully engineered few-shot examples and post-processing can significantly improve extraction quality with small models, even without task-specific fine-tuning.

Keywords

Information Extraction, Zero-Shot Learning, Few-Shot Prompting, Small Language Models, JSON Schema, Spanish NLP

1. Introduction

The GenSIE 2026 shared task, part of the IberLEF 2026 evaluation campaign [1, 2] introduces a challenging benchmark for evaluating systems that extract structured, nested JSON from general-domain Spanish texts. A key characteristic is the *zero-shot schema* challenge: the extraction schema is provided only at inference time, forcing systems to generalize to unseen structures rather than overfitting to fixed entity types or relation schemas. This builds on a growing body of work in schema-guided information extraction[3]. The task explicitly targets small, open-weight language models (sub-14B parameters), emphasizing inference-time techniques over scale-based solutions.

My submission takes a pragmatic approach to this challenge. I argue that for small models, the quality of the prompt and the robustness of post-processing matter as much as (if not more than) the choice of base model. My core insight is that small models benefit enormously from *examples*: showing the model what good output looks like via relevant synthetic few-shot examples, combined with explicit instructions for entity ordering, systematic scanning, and dynamic label guidance.

I originally designed three pipeline variants that explored the effect of prompt language (Spanish vs. English) and label generalization (hardcoded vs. dynamic schema reading). Because the official leaderboard reports only the best-performing pipeline per team, I focus this paper on the winning variant, `schema_dynamic`, which uses English prompts with dynamic schema reading. The pipeline uses a two-turn prompting strategy with validation-guided self-correction, schema-agnostic post-processing, and programmatic entity reordering to match gold-standard label priorities.

IberLEF 2026, September 2026, León, Spain
krishandheredia@gmail.com (K. Heredia)
0000-0002-1171-0150 (K. Heredia)

2. Related Work

2.1. Few-Shot Prompting for Information Extraction

The shift from task-specific extraction models to generative approaches has been driven by the rise of large language models (LLMs)[4]. While LLMs excel at generating structured output, they often struggle with structural reliability (invalid JSON) and zero-shot schema adherence[5]. Recent work has explored grammar-constrained decoding[6], Chain-of-Thought prompting[7], and self-consistency ensembling[8] to address these issues.

Few-shot prompting has emerged as a particularly effective strategy for IE without task-specific fine-tuning. GPT4NER[9] demonstrates that combining entity definitions, few-shot examples, and structured output formats transforms NER from sequence labeling into generation, achieving strong results on CoNLL2003 and OntoNotes5.0. PromptNER[10] further shows that integrating explicit entity type definitions alongside few-shot examples reduces classification ambiguity and improves cross-domain generalization. In the clinical domain, Naguib et al.[11] compare 13 auto-regressive models using prompting against 16 masked models using fine-tuning across English, French, and Spanish NER datasets, finding that prompt-based approaches achieve competitive performance while exhibiting lower environmental impact—supporting my emphasis on small, efficient models. Few-shot prompting with as few as 1–2 examples achieves competitive cross-lingual results for Spanish NER.

2.2. Schema-Guided and Zero-Shot Extraction

The GenSIE task's zero-shot schema challenge—where the extraction schema is provided only at inference time—has motivated several complementary approaches. KnowCoder[3] introduces a code-style schema representation that transforms schemas into Python classes, achieving substantial improvements over LLaMA2 in both few-shot and zero-shot settings. GoLLIE[12] demonstrates that providing explicit annotation guidelines alongside schemas significantly enhances zero-shot IE performance, with the model trained to follow detailed annotation rules for strong generalization to unseen schemas. GuideX[13] addresses the scalability challenge of manual guideline curation through structured synthetic data generation, showing that precise definitions and examples reduce error rates from ambiguous token boundaries. These findings motivate my combined approach of synthetic few-shot examples with dynamic label guidance derived from the schema.

2.3. Post-Processing and Structured Output

A growing body of work recognizes that raw LLM outputs require post-processing to ensure schema conformance. SLOT[14] formalizes the post-processing setting as transforming free-form text into structured formats according to a specified JSON schema, defining the task as model-agnostic and suitable for platform deployments serving diverse models. This directly supports my schema-agnostic post-processing pipeline. JSONSchemaBench and related constrained decoding frameworks demonstrate that schema-constrained output generation—whether through grammar-based decoding or post-hoc validation—is essential for reliable structured extraction.

2.4. Spanish Information Extraction

In the Spanish NLP community, the eHealth-KD shared tasks[15, 16] established foundational benchmarks for medical information extraction. GenSIE extends this tradition to a general-domain, zero-shot schema setting[2], with an emphasis on small models and sustainability—themes aligned with the broader movement toward efficient, deployable NLP systems. Recent work on Spanish clinical NER confirms that few-shot LLM approaches, augmented with external knowledge, perform on par with or exceed traditional BERT-based methods[11].

My work is most closely related to inference-time techniques for structured extraction. Rather than

fine-tuning or using massive models, I focus on prompt engineering and deterministic post-processing—an approach that is computationally efficient and model-agnostic.

3. Methodology

3.1. System Architecture

The pipeline follows a uniform architecture (Figure 1):

1. **Prompt Construction:** Build a system prompt with strict formatting rules and a user prompt with task-specific instructions, few-shot examples (for entity tasks), and the target JSON schema.
2. **LLM Inference:** Query an OpenAI-compatible inference server with temperature 0.5. I tested temperatures from 0 to 1 and selected 0.5 because it yielded the best extraction quality in my preliminary experiments.
3. **JSON Extraction:** Parse the model response using regex patterns for markdown code blocks and raw JSON objects.
4. **Post-Processing:** Apply schema-agnostic fixes including type normalization, entity filtering, entity reordering, and filling missing required fields.
5. **Validation & Self-Correction:** Validate against the target schema. If errors are found, append the output and correction instructions to the conversation history and retry (up to 3 attempts).

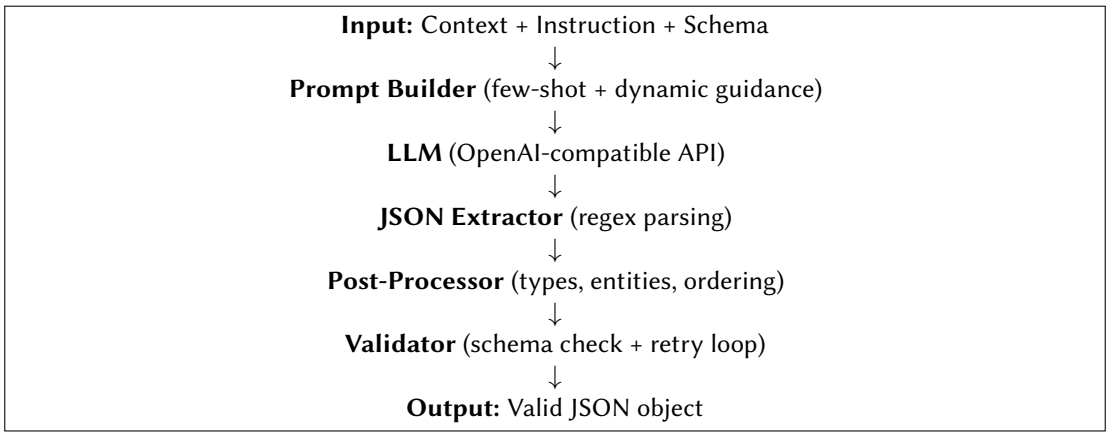

Figure 1: System architecture of the `schema_dynamic` pipeline.

Listing 1 shows the core inference loop shared by all pipelines.

Listing 1: Core inference loop (`schema_dynamic.py`)

```
def run(self, task: Task, model: str) -> Dict[str, Any]:
    system_prompt = self._build_system_prompt()
    user_prompt = self._build_user_prompt(task)
    messages = [
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": user_prompt},
    ]
    return self._run_with_retries(task, model, messages)


def _run_with_retries(self, task, model, messages):
    for attempt in range(self.max_retries):
        response = self.client.chat.completions.create(
```

```
            model=model, messages=messages,
                temperature=self.temperature
        )
        content = response.choices[0].message.content or ""
        json_str = self._extract_json(content)
        result = json.loads(json_str)
        result = self._post_process(result, task.target_schema,
            task)
        errors = self._validate_against_schema(result,
            task.target_schema)
        if not errors:
            return result
        if attempt < self.max_retries - 1:
            messages.append({"role": "assistant", "content":
                content})
            messages.append({"role": "user",
                "content": f"Fix errors: {errors}"})
    return result  # best-effort on final attempt
```

3.2. The `schema_dynamic` Pipeline

The submitted pipeline, `schema_dynamic`, uses English prompts with synthetic few-shot examples and dynamic label guidance. Its defining feature is that it reads allowed labels directly from the schema enum at runtime rather than relying on hardcoded entity labels. This allows the same pipeline to generalize to schemas with unseen entity types, which is the core challenge of the GenSIE task. All other design choices—two-turn prompting, validation-guided self-correction, schema-agnostic post-processing, and programmatic entity reordering by label priority—support this goal.

3.3. Prompt Engineering

For entity extraction tasks, I use a focused prompt with six key elements informed by recent few-shot IE literature. GPT4NER [9] demonstrates that combining entity definitions, few-shot examples, and structured output formats transforms NER from sequence labeling into generation, achieving strong results on standard benchmarks. PromptNER [10] further shows that integrating explicit entity type definitions alongside few-shot examples reduces classification ambiguity and improves cross-domain generalization. As few as 1–2 examples achieve competitive cross-lingual results for Spanish NER. GoLLIE [12] establishes that providing explicit annotation guidelines alongside schemas significantly enhances zero-shot IE performance. Finally, Xiao et al. [17] provide evidence that hybrid strategies combining type definitions and few-shot examples yield optimal performance for small models. Drawing on these findings, my prompts incorporate:

1. **Few-shot examples:** Two synthetic examples showing correct entity extraction with the expected JSON format.
2. **Dynamic label cheat-sheet:** Maps keywords from the task instruction to entity labels (e.g., "desarrollador" → PERSON, "empresa" → ORGANIZATION).
3. **Explicit ordering instruction:** Entities must be listed in the order PERSON, ORGANIZATION, LOCATION, DATE, EVENT, MISCELLANEOUS, matching the gold standard's label priority.
4. **Systematic scanning:** Step-by-step scanning instructions for each entity type, encouraging comprehensive coverage.
5. **Allowed labels:** A comma-separated list of permitted labels from the schema enum.
6. **Output template:** A minimal JSON template showing the expected structure.

Listing 2 shows the two synthetic few-shot examples used in the entity prompt.

Listing 2: Few-shot examples for entity extraction (`constants.py`)

```
EXAMPLES:

Example 1:
Instruction: Extract the authors, publishers, and book titles
    mentioned.
Text: "Gabriel Garcia Marquez published 'One Hundred Years of
    Solitude'
        with Editorial Sudamericana in 1967."
Output:
{"entities":[{"text":"Gabriel Garcia Marquez","label":"PERSON"},
  {"text":"1967","label":"DATE"},
  {"text":"Editorial Sudamericana","label":"ORGANIZATION"},
  {"text":"One Hundred Years of Solitude","label":"MISCELLANEOUS"}]}

Example 2:
Instruction: Identify city names, countries, and festivals from the
    text.
Text: "The Cannes Film Festival is held every year in France, near
    Nice."
Output:
{"entities":[{"text":"Cannes","label":"LOCATION"},
  {"text":"France","label":"LOCATION"},
  {"text":"Nice","label":"LOCATION"},
  {"text":"Cannes Film Festival","label":"EVENT"}]}
```

For non-entity structured tasks, I use a standard prompt with a schema-derived template, enum value lists, and extraction rules.

3.4. Post-Processing

Recent work emphasizes that robust post-processing is essential for reliable structured extraction from LLMs. SLOT [14] formalizes the post-processing setting as transforming free-form text into structured formats according to a JSON schema, defining the task as model-agnostic and suitable for diverse platform deployments. JSONSchemaBench [18] demonstrates that schema-constrained output—whether through grammar-based decoding or post-hoc validation—consistently improves downstream task performance. The validate-repair-retry pattern, where generated JSON is validated, errors are fed back for correction, and retries are capped, is widely recommended as the workhorse pattern for production LLM pipelines [19, 20]. Drawing on these findings, my post-processing pipeline is fully schema-agnostic and operates on the parsed JSON before validation:

- **Type normalization:** Converts string representations to integers, floats, and booleans. Normalizes enum values via case- insensitive matching and semantic mappings (e.g., "leve" → MILD, "común" → HIGH).
- **Entity filtering:** Removes entities not grounded in the source text, filters structural phrases, and deduplicates by text+label.
- **Entity reordering:** Sorts entities by label priority (PERSON → ORGANIZATION → LOCATION → DATE → EVENT → MISCELLANEOUS) and by first occurrence in the source text.
- **Missing field handling:** Fills missing required fields with sensible defaults (empty arrays, null for optional fields).
- **Property stripping:** Removes unknown properties not defined in the schema to prevent hallucinated fields.

Listing 3 shows the deterministic post-processing pipeline applied to every LLM output.

Listing 3: Post-processing pipeline (`post_processing.py`)

```
def _post_process(self, data, schema, task):
    # 1. Remove properties not in the schema
    data = self._strip_unknown_properties(data, schema, schema)
    # 2. Normalize types (enums, booleans, numbers, arrays)
    data = self._fix_types_recursive(data, schema, schema)
    # 3. Remove entities not grounded in source text; deduplicate
    data = self._filter_entities_if_present(
        data, schema, schema, task.input_text)
    # 4. Sort entities by label priority then text position
    data = self._reorder_entities_if_present(
        data, schema, schema, task.input_text)
    # 5. Replace null arrays with empty lists
    data = self._ensure_arrays_not_null(data, schema, schema)
    # 6. Fill missing required fields with defaults
    data = self._fill_missing_required(data, schema, schema)
    return data
```

3.5. Validation and Self-Correction

Deterministic validation with feedback-driven correction has emerged as a robust alternative to multi-model verification schemes. A three-stage validation pipeline—syntactic (JSON schema), semantic (custom rules), and textual-fidelity (source-text grounding)—achieves comparable precision to collaborative or verifier-augmented approaches. Self-correction patterns for structured output failures are well documented: approximately 95% of correctable failures resolve on the second attempt, with diminishing returns thereafter, and combining LLM self-correction with schema validators yields high accuracy at the cost of modest extra token overhead (10–20% per call). Drawing on these findings, my validation and self-correction pipeline operates as follows.

After post-processing, I validate the output against the target JSON schema using recursive type checking, enum validation, and required field verification. If validation fails, I append the erroneous output and a correction prompt to the conversation history and request a revised JSON. This self-correction loop runs for up to 3 attempts. On the final attempt, I return the best-effort output even if validation errors remain, since partial credit is preferable to a zero score.

4. Experimental Setup

4.1. Data

I evaluate on the official GenSIE 2026 test set[21]. The development set contains 271 silver instances across diverse domains (legal, scientific, news, technical) and eight domains, while the test set contains 145 gold instances with unseen schemas to evaluate zero-shot generalization. The official evaluation was run on 125 instances after excluding 20 instances whose JSON schemas could not be reliably evaluated on the inference stack.

4.2. Models

The task enforces a no-training policy: participants cannot fine-tune models or load external adapters. All systems connect to an OpenAI- compatible inference server hosted by the organizers. The official evaluation was conducted against two published models: Gemma 4 E4B and Qwen3-14B. I developed the pipelines using a local dry run with Llama 3.2 (3B parameters), but the reported results are on the

official evaluation models. Recent work on Spanish clinical NER confirms that few-shot prompting with small language models achieves competitive performance on Spanish extraction tasks[11], supporting my choice of a prompting-based approach without task-specific fine-tuning.

4.3. Evaluation Metrics

GenSIE uses a Flattened Schema Scoring metric[2]. JSON objects are flattened into key-value pairs, and values are scored by type: rigid types (numbers, dates, enums) require exact matches; free-text fields use a hybrid of cosine similarity and lexical overlap. Precision, Recall, and F1 are computed from True Positive Scores.

5. Results

Table 1 reports the official test-set results for my `schema_dynamic` pipeline on the 125 scored instances. The baseline F1 scores for the reference zero-shot pipeline were 0.7895 on Gemma 4 E4B and 0.7805 on Qwen3-14B.

Table 1
Official test-set results for `schema_dynamic` (125 scored instances).

Model	F1	Gap closed
Gemma 4 E4B	0.8142	11.73%
Qwen3-14B	0.8076	12.35%
Average	—	**12.02%**

The `schema_dynamic` pipeline achieved F1 scores of 0.8142 and 0.8076 on Gemma 4 E4B and Qwen3-14B respectively, corresponding to a 12.0% average gap closed over the reference baseline. These results were statistically significant ($p < 0.001$, paired bootstrap) on both models. The consistent improvement across both model families suggests that the pipeline generalizes well across different small-model architectures.

Table 2 reports the efficiency of the `schema_dynamic` pipeline. The single-call design keeps token usage well below the 32K-token soft budget.

Table 2
Efficiency of the `schema_dynamic` pipeline.

Metric	Value
Avg. tokens per instance	∼4.3K
Gap closed per 1K tokens	0.0288
LLM calls per instance	1 (plus validation retries)

6. Discussion

6.1. Overall Performance and Effectiveness of Few-Shot Prompting

The `schema_dynamic` pipeline closed 12.0% of the baseline-to-perfect gap, demonstrating that carefully engineered few-shot prompting combined with schema-agnostic post-processing can meaningfully improve extraction quality for small models on zero-shot schemas. The absolute F1 gains are modest—+2.47 and +2.71 points over the Gemma and Qwen baselines respectively—but they are meaningful given that the baselines were already near 0.79, leaving only about 21% of error headroom. Closing 12% of that remaining gap with a single-call, no-training approach is a substantive result.

The consistent improvement across both Gemma and Qwen model families suggests that the approach is model-agnostic. This is important for deployment scenarios where the specific model may vary, and it aligns with findings that few-shot prompting with entity definitions and structured output formats is an effective strategy for small models[9, 10]. The choice of prompt language also matters for multilingual models[22], a question I return to below.

6.2. Prompt Language Design

I chose English prompts for `schema_dynamic` based on prior work showing that non-native (English) prompts can outperform native-language prompts for some multilingual models. Kmainasi et al.[22] found that English prompts outperformed native-language prompts on average across 197 experiments with Arabic datasets, and Etxaniz et al.[23] showed that a self-translate strategy (translate to English, then reason) consistently outperforms direct non-English inference on several multilingual models. This literature motivated the decision to use English prompts for a Spanish extraction task, though it is important to note that the official evaluation does not include a controlled ablation of prompt language. The observed performance therefore reflects the combined effect of English prompting and dynamic label derivation, and future work could isolate these factors more cleanly.

6.3. Dynamic Schema Support

The central design choice of `schema_dynamic` is that it reads allowed labels dynamically from the schema enum at runtime, rather than relying on a fixed, hardcoded label set. This design directly addresses the zero-shot schema challenge: when a schema with unfamiliar labels appears at inference time, the pipeline can still provide correct label guidance to the model. By deriving labels from the schema, the pipeline removes the risk of mismatches between the labels a model is instructed to use and the labels actually defined in the schema. This finding is consistent with broader evidence that schema-aware prompting improves zero-shot extraction performance[12, 3].

6.4. Post-Processing and Validation-Guided Self-Correction

The post-processing pipeline operates deterministically and schema-agnostically on every LLM output before validation. Type normalization, entity filtering, entity reordering, missing-field handling, and property stripping together address common failure modes without encoding schema-specific rules. The validation-guided self-correction loop provides a lightweight mechanism for recovering from schema violations: if validation fails, the erroneous output and an error message are appended to the conversation history and the model retries, up to three times. The efficiency results in Table 2 indicate that most instances resolve in a single call and that retries add modest token overhead when they occur.

This architecture—free generation followed by post-hoc validation and correction—embodies the "reason free, constrain late" principle advocated by recent work on structured output from small language models[24]. By allowing the model to generate freely without grammar constraints and enforcing schema conformance in post-processing, the pipeline avoids the "constraint tax" that can reduce answer accuracy when strict decoding constraints are applied too early. This is particularly relevant given that grammar-constrained decoding approaches in the shared task encountered structural failures on certain schema constructs that rendered them unscoreable.

6.5. Token Efficiency and Computational Cost

The `schema_dynamic` pipeline achieves 0.0288 gap closed per 1K tokens, placing it among the most efficient approaches in the shared task. This efficiency is a direct consequence of the single-call design: each instance requires exactly one primary LLM call, with validation retries only when needed. There is no retrieval, ensemble voting, or multi-extractor fan-out. For schema-guided extraction with small models, these results suggest that a well-engineered single call can be both token-efficient and competitive in extraction quality. This has practical implications for deployment, where lower token

consumption translates to lower latency and cost, which is critical for edge scenarios that the GenSIE task targets.

6.6. Limitations and Future Work

Several limitations should be noted. The pipeline was evaluated only on the two published models (Gemma 4 E4B and Qwen3 14B), so performance on other model families or sizes within the sub-14B range may vary. I did not perform a controlled ablation of prompt language, so the relative contribution of English prompting and dynamic label derivation cannot be isolated from the observed scores. The few-shot examples are synthetic and static, and do not adapt to the specific schema or text at hand. Future work could explore dynamic few-shot example selection for potentially better guidance. The post-processing pipeline is entirely rule-based; a learned correction component could improve accuracy at the cost of requiring training data[14]. Self-consistency ensembling was not explored due to token budget considerations, but remains an interesting direction for future work[8]. Finally, the zero-shot schema setting is challenging but realistic for deployment; future work could explore hybrid approaches that combine schema-agnostic prompting with lightweight schema-specific adaptations.

7. Conclusion

I presented a few-shot prompting pipeline for the GenSIE 2026 shared task. The `schema_dynamic` approach emphasizes English-language prompt engineering, dynamic label derivation from the schema, schema-agnostic post-processing, and validation-guided self-correction as complementary strategies for improving small-model performance on zero-shot structured extraction. All code is open-sourced under the MIT license and available at https://github.com/krishanheredia97/krishan-heredia-gensie-submission.

Acknowledgments

I thank the GenSIE 2026 organizers for providing the task definition, datasets, and evaluation infrastructure. This research was conducted independently without institutional funding.

Declaration on Generative AI

During the preparation of this work, the author used Kimi K2.6 and Kimi K2.7 for code assistance, debugging, writing the Discussion section, and grammar and spelling check. After using these tool(s)/service(s), the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] 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.

[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] Z. Li, Y. Zeng, Y. Zuo, et al., KnowCoder: Coding structured knowledge into LLMs for universal IE, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, ACL, 2024.

[4] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Sun, S. Liu, P. Liu, J. Nie, J.-R. Wen, A survey of large language models, arXiv preprint arXiv:2303.18223 (2023).
[5] C. Shorten, C. Pierse, T. B. Smith, E. Cardenas, A. Sharma, J. Trengrove, B. van Luijt, Structuredrag: Json response formatting with large language models, arXiv preprint arXiv:2408.11061 (2024).
[6] B. T. Willard, R. Louf, Efficient guided generation for large language models, in: arXiv preprint arXiv:2307.09702, 2023.
[7] J. Wei, et al., Chain-of-thought prompting elicits reasoning in large language models, Advances in Neural Information Processing Systems 35 (2022).
[8] X. Wang, et al., Self-consistency improves chain of thought reasoning in language models, arXiv preprint arXiv:2201.11903 (2022).
[9] Y. Zhao, X. Zhong, E. Cambria, J. C. Rajapakse, GPT4NER: Large language models for few-shot named entity recognition, arXiv preprint arXiv:1810.06818 (2018).
[10] D. Ashok, Z. C. Lipton, PromptNER: Prompting for named entity recognition, in: arXiv preprint arXiv:2305.15444, 2023.
[11] M. Naguib, X. Tannier, A. Névéol, Few-shot clinical entity recognition in English, French and Spanish, in: Findings of the Association for Computational Linguistics: EMNLP 2024, ACL, 2024.
[12] O. Sainz, I. García-Ferrero, R. Agerri, et al., GoLLIE: Annotation guidelines improve zero-shot information extraction, in: Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024), 2024. ArXiv:2310.03668.
[13] N. De La Fuente, O. Sainz, et al., GuideX: Guided synthetic data generation for zero-shot information extraction, in: Findings of the Association for Computational Linguistics: ACL 2025, ACL, 2025.
[14] D. Y.-B. Wang, Z. Shen, S. S. Mishra, Z. Xu, Y. Teng, H. Ding, SLOT: Structuring the output of large language models, arXiv preprint arXiv:2505.04016 (2025).
[15] A. Piad-Morffis, Y. Gutiérrez, S. Estévez-Velarde, Y. Almeida-Cruz, A. Montoyo, R. Muñoz, Analysis of ehealth knowledge discovery systems in the TASS 2018 workshop, Procesamiento del Lenguaje Natural 62 (2019) 13–20. doi:10.26342/2019-62-1.
[16] A. Piad-Morffis, Y. Gutiérrez, J. P. Consuegra-Ayala, S. Estevez-Velarde, Y. Almeida-Cruz, R. Muñoz, A. Montoyo, Overview of the eHealth knowledge discovery challenge at IberLEF 2019, in: Proceedings of IberLEF 2019, 2019. https://scholar.archive.org/work/461857e9-d292-4dd6-9395-80db89ae5b89.
[17] Y. Xiao, et al., Advancing few-shot named entity recognition with large language models, Applied Sciences 15 (2025) 3838.
[18] S. Geng, H. Cooper, M. Moskal, S. Jenkins, J. Berman, N. Ranchin, R. West, E. Horvitz, H. Nori, JSONSchemaBench: A rigorous benchmark of structured outputs for language models, arXiv preprint arXiv:2501.10868 (2025).
[19] Tetrate, LLM output parsing and structured generation guide, https://tetrate.io/learn/ai/llm-output-parsing-structured-generation, 2025. Blog post.
[20] C. Wilkins, LLM structured outputs: Schema validation for real pipelines, https://collinwilkins.com/articles/structured-output, 2026. Personal blog.
[21] GenSIE 2026: General-purpose schema-guided information extraction, https://uhgia.org/gensie/, 2026. Official task website.
[22] M. B. Kmainasi, R. Khan, A. E. Shahroor, B. Bendou, M. Hasanain, F. Alam, Native vs non-native language prompting: A comparative analysis, arXiv preprint arXiv:2409.07054 (2024).
[23] J. Etxaniz, G. Azkune, A. Soroa, O. Lopez de Lacalle, M. Artetxe, Do multilingual language models think better in english?, in: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers), 2024, pp. 491–506. URL: https://aclanthology.org/2024.naacl-short.46/.
[24] J. Ray, The constraint tax: Measuring validity-correctness tradeoffs in structured outputs for small language models, arXiv preprint arXiv:2605.26128 (2026). URL: https://arxiv.org/abs/2605.26128.