Gradiant at IberLEF 2026 GenSIE: Taming SLMs for Information Extraction

Álvaro Bueno Sáez[1,*,†], David Sueiro-Durán[1,†] and Héctor Cerezo-Costas[1,†]

[1]Fundación Centro Tecnolóxico de Telecomunicacións de Galicia (GRADIANT), Vigo, Spain

Abstract

This paper describes the Gradiant NLP Team submission to the GenSIE task at IberLEF 2026. This task evaluates SLMs for zero-shot generalized structured information extraction from texts. We propose three inference pipelines model-agnostic that no require fine-tuning and operate over a 60 seconds time limit restriction per instance. The *stable* pipeline combines two steps analysis entity generation followed by text markup with finally a extraction loop and a verification pass. The *experimental* pipeline classifies the schema fields into five categories and assigns a dedicated extraction strategy to each category. The *limited* pipeline extends the *experimental* one and added a time gated chain-of-thought reasoning, activating reasoning, *high* or *low*, based on the remaining time. We evaluate all pipelines across six models ranging from $1.7B$ to $8B$ parameters, including open and closed models. The *stable* pipeline consistently outperforms the provided baseline across all models and datasets, achieving up to F_1 of 0.76 on the *starter* set (Gemma4:e4b) and F_1 of 0.79 on the *dev* set (Gemma4:e4b, limited pipeline). A key finding is that recent open-weight SLMs such as Gemma4:e4b and Qwen3-4B match or surpass commercial GPT-4o-mini on this task, suggesting that ad-hoc inference engineering can offset the gap between small and large models. Our implementation is publicly available at https://github.com/Gradiant/SI-NLP-GenSIE-2026

Keywords

Small Language Models, Structured Information Extraction, GenSIE 2026, IberLEF

1. Introduction

Information Extraction (IE) [1] is the classical NLP task of extracting structured information from raw content. Traditionally focused on entity extraction or relation extraction, the zero-shot capability of Large Language Models (LLMs) allows to extend the IE task to zero-shot problems, without relying on extensive training corpora. Those models are also well adapted to generate structured information, as many interfaces that operate with these models implement functionality to parse the output using schemas.

Output schemas enforce valid output generation, adding restrictions to guarantee that the output can be correctly parsed and mitigating hallucinations, or using other words, structured outputs behave as hard constraints on the generation process [2] (e.g., when asked to categorize content, a generative model could tag with one of the requested categories or hallucinate answering with a made-up new one). This is especially relevant when using Small Language Models (SLMs).

GenSIE [3] is a task organized within the IberLEF 2026 congress [4] with the aim of advancing in the use of SLMs for generative IE. This paper describes our approach to the task, in which we propose three strategies that use multiple steps combining self-consistency, problem decomposition, and balancing computation factorized by the remaining time.

This paper is organized as follows: section 2 presents state of the art in generated information extraction with language models, section 3 presents the problem to solve, section 4 describes our solution to the problem, section 5 explains the evaluation framework, section 6 presents the results of the methods implemented. Finally, sections 7 and 8 describe interesting insights extracted from the participation in the task and conclude the paper.

IberLEF 2026, September 2026, León, Spain

*Corresponding author.

†These authors contributed equally.

abueno@gradiant.org (Á. B. Sáez); dsueiro@gradiant.org (D. Sueiro-Durán); hcerezo@gradiant.org (H. Cerezo-Costas)

0009-0006-9199-972X (Á. B. Sáez); 0009-0008-0937-5462 (D. Sueiro-Durán); 0000-0003-2813-2462 (H. Cerezo-Costas)

2. Background

Similar to the GenSIE task, InstructIE [5] generates a synthetic dataset using a guided schema per topic. This idea is refined in IEPile, which is a large dataset [6] created to benchmark language models for guided information extraction using existing IE datasets. However, the scope is more restricted as it focuses only on three types of tasks, Entity Extraction, Relation Extraction and Event Extraction.

LLMs are very capable of solving these tasks. Typically, those benchmarks are tested with state-of-the-art or commercial models, but the open question is how smaller and energy-efficient models perform in the same tasks. Small Language Models (SLMs) with less than 10B parameters have proven very capable when solving mathematical problems [7], reasoning [8, 9], or in tool calling [10] , but they have not been profusely tested in information extraction.

Language models could hallucinate in IE tasks or can produce invalid outputs when structured outputs are requested. This is more common in SLMs. One way to mitigate this is by using Grammar Constraint Decoding (GKD) [11] in which the valid tokens for each step are limited by grammar rules. This enforces the generation of only valid outputs. Nevertheless, GKD has its limitations, such as token-boundary misalignment or complex schemas that need more than a syntactic validation [12]. In those cases, employing extra steps for validation seems to be more beneficial and practical than using complex GKD implementations. Another technique, self-consistency, which executes multiple times the same task to pick the most repeated answer, could help even in reasoning tasks [13] although it is consequently more time-consuming. Annotation guidelines [14] have proven beneficial in some scenarios, especially when the models have been finetuned to follow them. In [15], they create schemas using two techniques: schema retrieval from a pool of existing schemas and schema generation for those tasks never seen before.

The GenSIE task is different from previous benchmarks. With a strong focus on small language models and generalization across tasks and models, it poses a harder challenge by requiring strict zero-shot adherence to dynamic schemas, robust structured output generation, and reliable extraction across heterogeneous domains without relying on large-scale model capacity or task-specific fine-tuning.

3. Task Analysis

The GenSIE task, explained in depth in [3], focuses on the evaluation of SLMs for IE with zero-shot inference. The decision to only evaluate SLMs is based on efficiency and sustainability criteria, and also because SLMs require clever engineering to earn the same results as LLMs. Also, the IE presented in this task consists of structured and nested data from plain text, a question, and the schema of the answer. The schema can request to fill just one or multiple fields, either simple or nested. Those fields could imply verbatim extraction from the raw text, text categorization from a limited set of categories, text generation (e.g., generating a question), entity extraction, and nearly any other conceivable task.

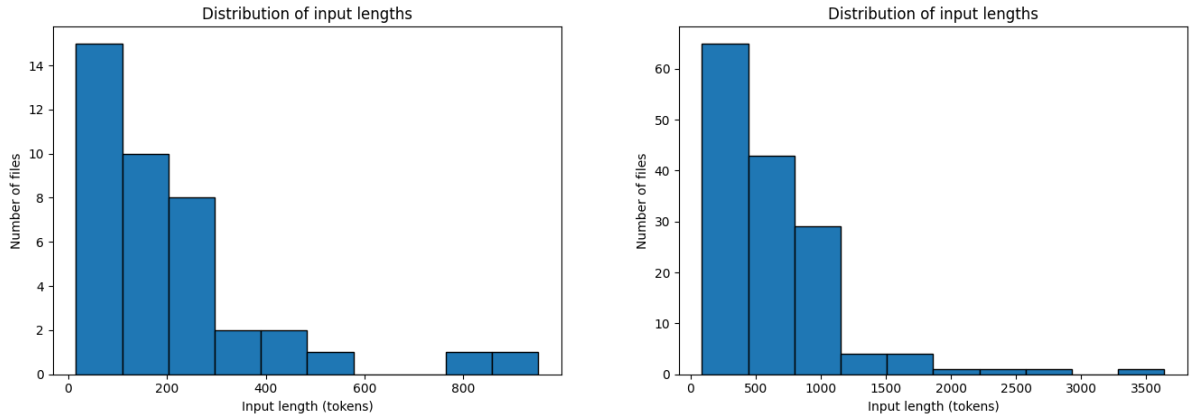
Models must adapt in real time to the schemas without relying on fine-tuning. Furthermore, solutions must be model agnostic, taking into account the assorted models with different capabilities (tool calling, reasoning, etc.). They interact with the models through API calls and must adapt to model capabilities without access to model internals or exhaustive output analysis. Time is also a limiting factor in this benchmark, as each task must be solved in one minute maximum.

Due to the variability of the outputs and their types, finding a suitable metric that evaluates the performance of the different strategies is very challenging. Not all the answer schema types are equally difficult: binary outputs have only two possible values, categorization can have multiple, whereas nested arrays have multiple fields that must be extracted correctly. The metric for this task was calculated depending on the output type. The different output objects were transformed to a type more suitable for evaluation: JSON outputs are flattened, rigid types have binary precision, and free text is measured with semantic embeddings. With these transformations, the True Positive Score (TPS) is calculated as well as accuracy, recall, and F1 score.

3.1. Dataset analysis

Two datasets have been released by the organizers to help with code implementation: *starter* and *dev*, one with 49 entries and the second one with 149. Before starting to develop our strategies, we performed an analysis of the data, evaluating their length, type of data requested, and their frequency in the dataset, and complexity.

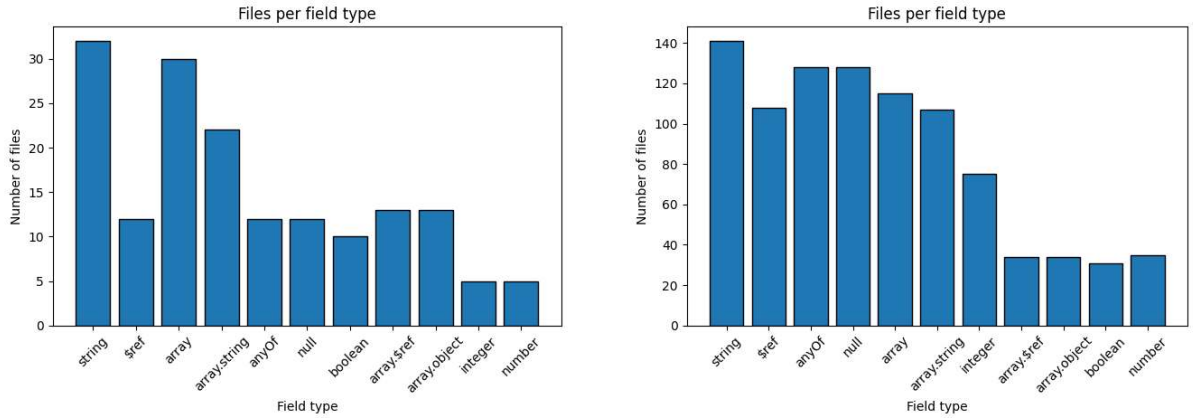
First, in Figure 1 we plotted the distribution of input lengths (the raw text), which shows how most of the file contains lower lengths (in tokens). There are some outliers with a very high amount of tokens. This behaviour happens both with the *dev* dataset as well as the *starter* dataset.



(a) starter (b) dev

Figure 1: Token length analysis in *starter* and *dev* datasets

Secondly, we evaluated the number of types in each file in Figure 2



(a) starter (b) dev

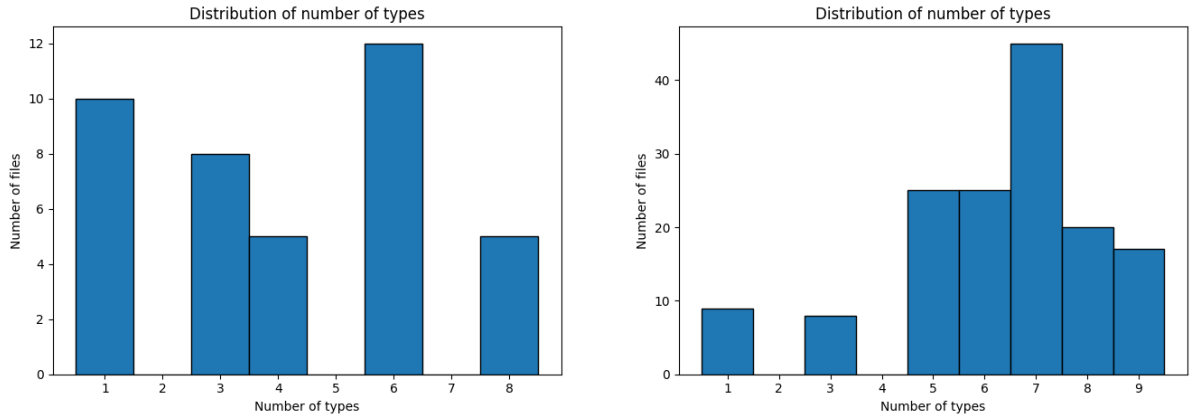
Figure 2: Number of appearances of each data type in *starter* and *dev* datasets

The figure above displays the 12 types presented in the datasets and how many files require that type of answer. Of all these 12 types, some considerations should be commented on:

1. "$ref" label: it means that the field contains fixed strings.
2. "array $ref": it means that the field contains an array of fixed strings.
3. anyOf: this means that the field contains not only one type of data, but multiple types of data.

Under these considerations, the main conclusion of Fig. 2 is that most of the data types are arrays and strings, but also complex structures of arrays that are much more difficult to extract. The next question for this analysis is how the data type distribution is in each file.

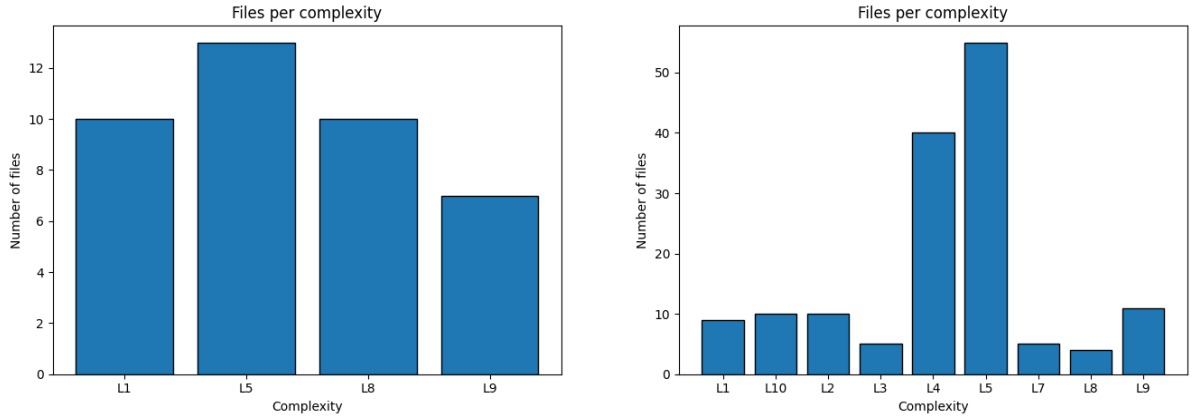
Third, the distribution of the number of types is plotted in 3, which shows that the *dev* dataset has more files that require the extraction of more data types than in the *starter* dataset.



(a) starter (b) dev

Figure 3: Number of types in each file in the starter and *dev* dataset.

Finally, it is interesting to analyze also if this information about length and types correlates with the associated complexity levels. The task defines 10 levels of complexity but not all levels are present in both datasets, as is shown in Figure 4. The starter dataset only contains 4 levels (L1, L5, L8, L9) whereas the dev dataset contains 9 levels (only the L6 is missing).



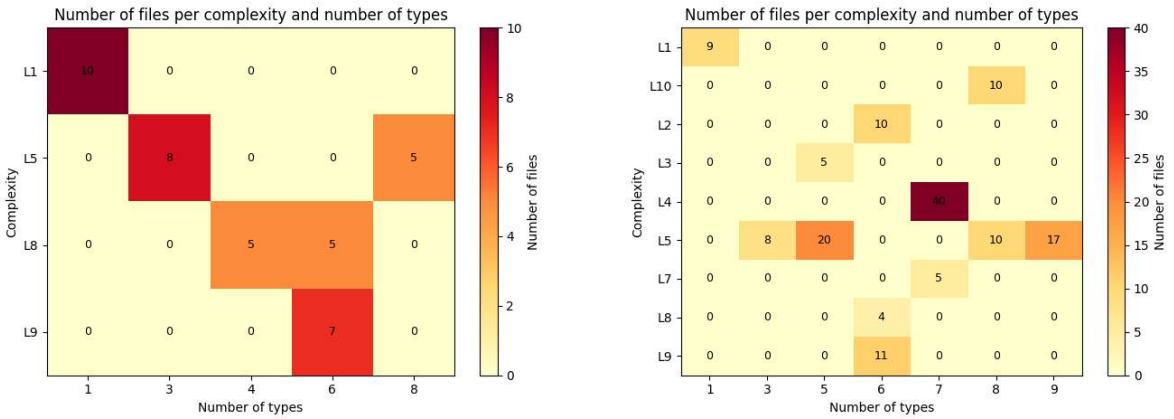
(a) starter (b) dev

Figure 4: Number of complexity levels in the *starter* and *dev* datasets.

Also, to understand whether the complexity in each schema concerning the number of types, the Figure 5 allows us to know that with a heat map.

In figure 5 many observations could be made:

1. In the *starter* dataset (a), it seems that the complexity is not related to the number of types because low levels like L5 contain 5 files with 8 different types, and high levels like L9 contain only 6 different types.
2. Also in the *starter* dataset (a), it seems that the files are well distributed along all complexities despite this dataset only contains 4/10 levels of complexity.
3. In the *dev* dataset (b), the lack of relation between complexity and the number of types is confirmed. The lower complexity (L1) seems to be the simplest one (with only one type), but the rest of the complexities are not related to the types: L10 always has 8 types, L4 has 7 types, etc.
4. Finally, in the *dev* dataset (b), the distribution is finally biased to L4 and L5 complexities.



(a) starter (b) dev

Figure 5: Heatmaps with the number of types per complexity level

3.2. Test analysis

The test dataset has also been released to obtain the final scores. To understand the result that will be explained in section 6, an analysis of the test dataset should also be done to understand the differences or similarities with the dev and starter datasets.

In the same way as in the previous section, the analysis of length, types, and complexity will be made in the same order. First, in the figure 6, a growth of text length can be observed. While in 1a and 1b the maximum value was around 800 for started and 3500 for dev, now for test the maximum text length in tokens is around 10,000.

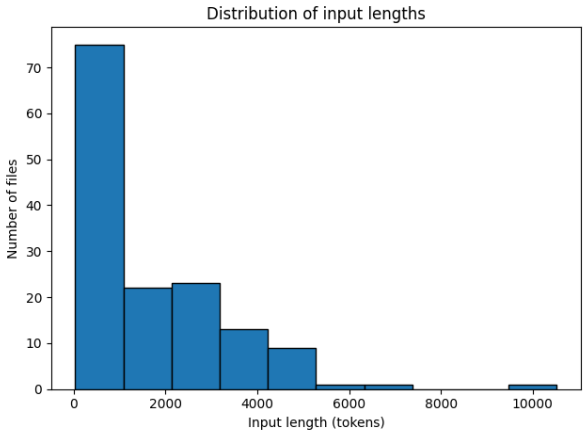


Figure 6: Token length analysis in test dataset

Also, Table 1 shows how the number of characters in each dataset has increased. In row 2, the growth has been calculated as the growth percentage for each dataset based on the previous dataset.

metric	starter	dev	test
mean char	878.6	2412.37	6262.10
growth (%)	0	174.57	159.58

Table 1

Mean and growth for character len in starter, dev and test

In figure 7, we can observe how the fields by field types maintain their distribution with respect to the dev dataset distribution. The only significant differences are in the field anyOf and null, which

change from 80 occurrences in both the dev dataset to 130 in the test datasets. This growth means that there are more null fields present in the test dataset, and also more fields with two or more different types.

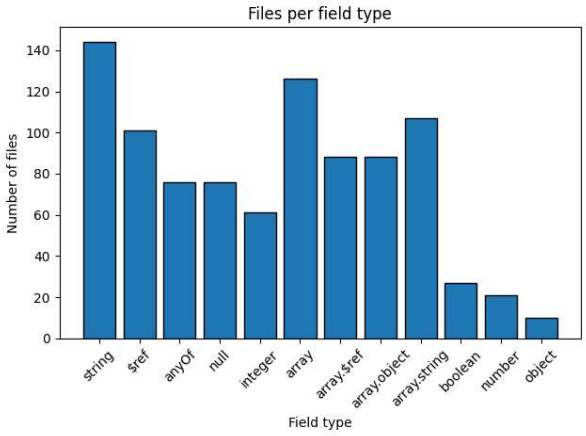


Figure 7: Number of appearances of each data type in test dataset

This growth in the number of types for each field can be confirmed by the figure 8, where the files with fields with 3 types have been increased from 2 to 10, and for fields with 4 types, there was also an increase from 25 to 50. While these are the main growths, we do not observe significant growth in the rest of the columns for table 2b and 7

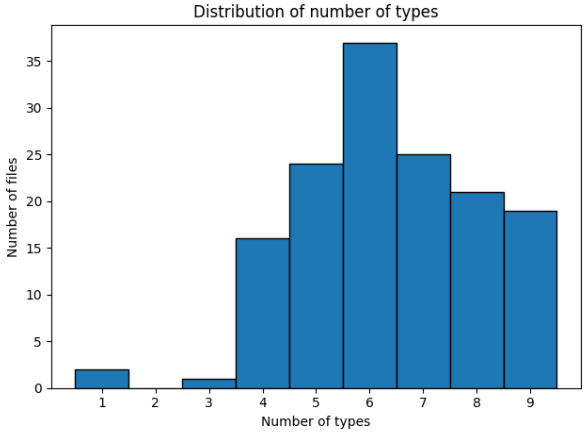


Figure 8: Number of types in each file in the test dataset

Finally, a study of complexity should also be done. Comparring table 4b and 9 we can observe the following evidences:

1. L1,L2, L4, L5 have been decreased.
2. L2,L3, L9 has a sligthly increaments.
3. L7 and L8 have a significant increase.
4. L6 is a new complexity never shown in previous datasets.
5. unknown complexity due to the changes in the format in which the complexity is written in the target schema description.

To understand the distribution of tasks with different complexities, in figure 10 we found aging the heatmap for complexities and number of types.

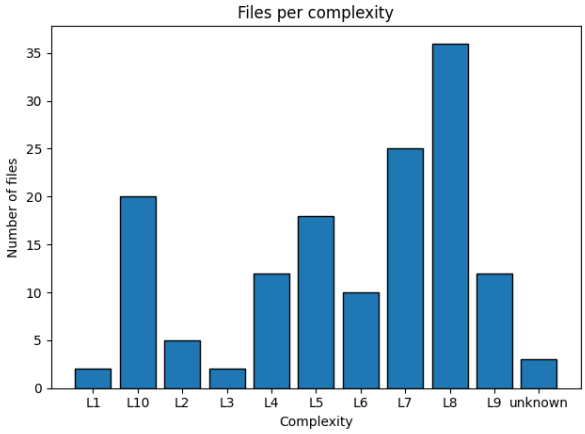


Figure 9: Number of complexity levels in the test dataset

In comparison with 5b, the distribution of complexities is not centralized on L4 now, being the dsitributión more plain. However, we can observe how complications are lowly represented in dev and are now represented in more files and with more different types.

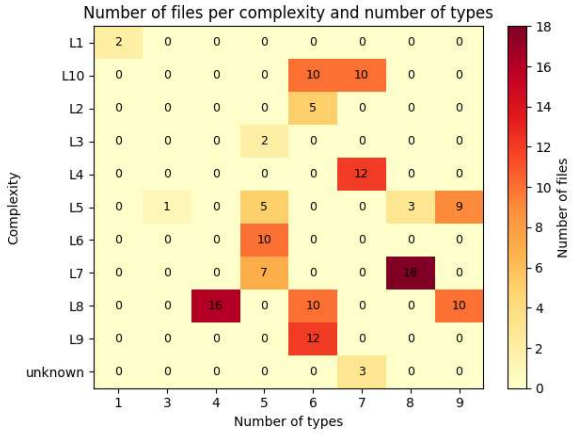


Figure 10: Heatmaps with the number of types per complexity level for test

In conclusion, there are several differences between the test dataset and the dev datasets, with the most significant being the length of texts. This difference is important because it could affect the behaviors of the models, considering their inference time implications. .

4. Methods

This section explains our three methods submitted to the GenSIE task. *Stable* pipeline that adds steps for analysis of the task and validation of the results with self-consistency, disabling the reasoning of LMs, *experimental* pipeline that includes strategy planning and reasoning based on the remaining budget time, and the *limited* pipeline that extends the *experimental* pipeline with dynamic activation of the reasoning capabilities of models.

4.1. Stable pipeline

This strategy implements three sequential phases: Analysis, Extraction, and Validation.

Analysis The objective of this phase is to analyse the input providing additional guidelines to the extraction. During this phase, it does not produce actual output following the schema but the *reasoning text* that will be added to the task description. It performs two sequential calls to the language model to avoid overloading it with instructions. The first call conditions the model to make a description of the entities defined in the problem schema, to provide concrete examples that can be found in the text and to list the detected entities. This information is used in the second call of the phase in which the entities requested are marked (although not extracted) in the original text. This step was found to be beneficial to extract verbatim text from the original content.

Extraction This phase makes successive language model calls using the augmented information of the task of the Analysis phase in order to extract the entities constrained to the schema requested. This process is repeated a maximum of 5 times when the remaining allotted time is higher than 30 seconds.

Validation and Merging This final phase is only executed if at least 17 seconds could be spent on the task before reaching timeout. This phase makes the language model to merge the extractions of the previous step into one unique response. This phase overloads the model with multiple operations (merging and deleting wrong responses). When this step is skipped due to closeness to the timeout, the first extraction of the Extraction phase is selected instead.

All the language model calls share the same hyperparameters (*temperature* $= 0.1$ and *reasoning_effort* $= None$ to avoid reasoning as it could be very time-consuming, even with small models). To disable reasoning with Qwen models, the */nothink* was added to the system prompt.

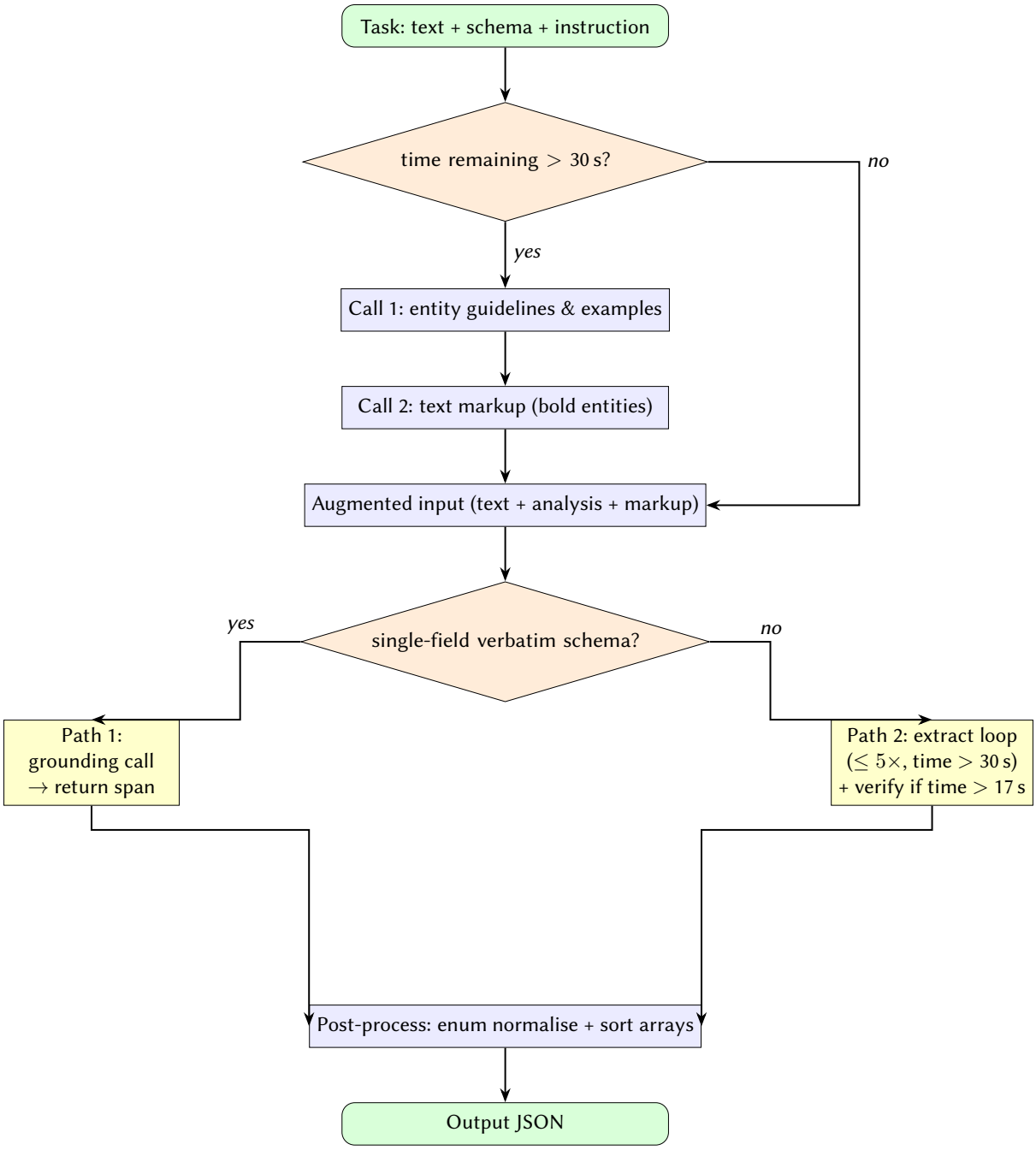


Figure 11: Execution flow of the *stable* pipeline (`StableAgent`). When more than 30 s remain, a two-call analysis phase enriches the input: the first call produces entity guidelines and concrete examples; the second copies the original text verbatim adding bold markers around target entities. The augmented input is then routed to one of two strategies. **Path 1** (single-field verbatim schemas): a grounding call locates the exact source span and returns it directly as the answer. **Path 2** (all other schemas): a self-consistent extraction loop runs up to five times while at least 30 s remain, followed by a verification and merging pass if at least 17 s are still available. All results are post-processed to fix enum case mismatches and sort array fields by schema-definition order.

4.2. Experimental pipeline

The experimental pipeline is implemented by ExperimentalAgent class which incorporates strategy planning and time and reasoning based execution as approaches to resolve the task.

Strategy planning. At the beginning of each execution, a plan is made for assigning each field to each specific method to be extracted. These fields are categorized by the planner into 5 different

categories, which are named the same as the strategy that implements the code for its inference. The categories are:

1. Direct: for answer with a direct anwer /(string or number)
2. Categorical: for answers with predetermined values.
3. Fixed entities: for a list of dicts (entities) with predeterminated values for classification.
4. Soft entities: for a list of entities with no particular values.
5. Complex: for complex structures that are not exactly entities.

With this classification, the planner establishes a path to run the execution, assigning the proportional time to each step. It is important to clarify that each step in the execution plan contains only one strategy, but one strategy could include many fields. For example, if a task contains three fields classified as direct and one as soft entities, the planner would make a plan with two steps: one for the three fields classified as direct, and the second for the other one.

Time resoning-based execution. In practice and due to a lack of time, only three strategies had been implemented. All entity strategies (soft and fixed) are processed by the same strategy, and also all direct strategies (direct and categorical). These strategies had been implemented to consider the time remaining to adjust the reasoning level of the model. This time estimation system takes into account the time that other calls to the LLM have taken to estimate the next execution time. Depending on the remaining time, each strategy will run its method for inference with more or less reasoning level (none, low, medium, or high). Also, each strategy has its own prompts and methods for resolving the task. For example, a direct strategy uses a simple prompt, while a complex strategy first asks for candidates or hints to improve the next call to the LLM.

4.3. Limited pipeline

The *limited* pipeline is implemented by the `ExperimentalAgentV2` class. It preserves the full architecture of the experimental pipeline: the field categorisation, the plan generation, and the strategy execution for each category (see Section 4.2 for details). The key novelty in this pipeline es the dynamic activation of the model's chain-of-thought reasoning for steps.

Reasoning categories. Only fields belonging to the categories `fixed_entities`, `soft_entities`, and `complex` trigger reasoning. These are multi-token structured outputs with entity lists and nested objects, where an additional reasoning pass has been found to reduce hallucinations. The categories `direct` and `categorical` are intentionally excluded because the map to simple extractions so the reasoning call effort is not justified in this case.

Time gate. Reasoning is activated only when the remaining clock time exceeds a threshold $\tau = 40\,\text{s}$ within the 60 s competition budget. It's a small limit due that the models with (`reasoning_effort="hight"`) take a long time to generate the output, therefore being an important point of possible timeouts This ensures that the slower reasoning calls do not push the pipeline to pass the instance time limit. Once elapsed time reduces the remaining budget below τ, the pipeline falls back to standard (non reasoning) inference for all the next steps. Each strategy step also receives a hard timeout of $\max(\text{remaining} - 2, 5)$ s, leaving a two second safety margin and a minimum floor of five seconds.

Model compatibility. The reasoning flag (`reasoning_effort="low"`) is only passed to models that expose a chain-of-thought interface: members of the Qwen3, QwQ, DeepSeek-R1, O1, O3, and O4 families. For other models the pipeline degrades gracefully to the standard `ExperimentalAgent` behaviour.

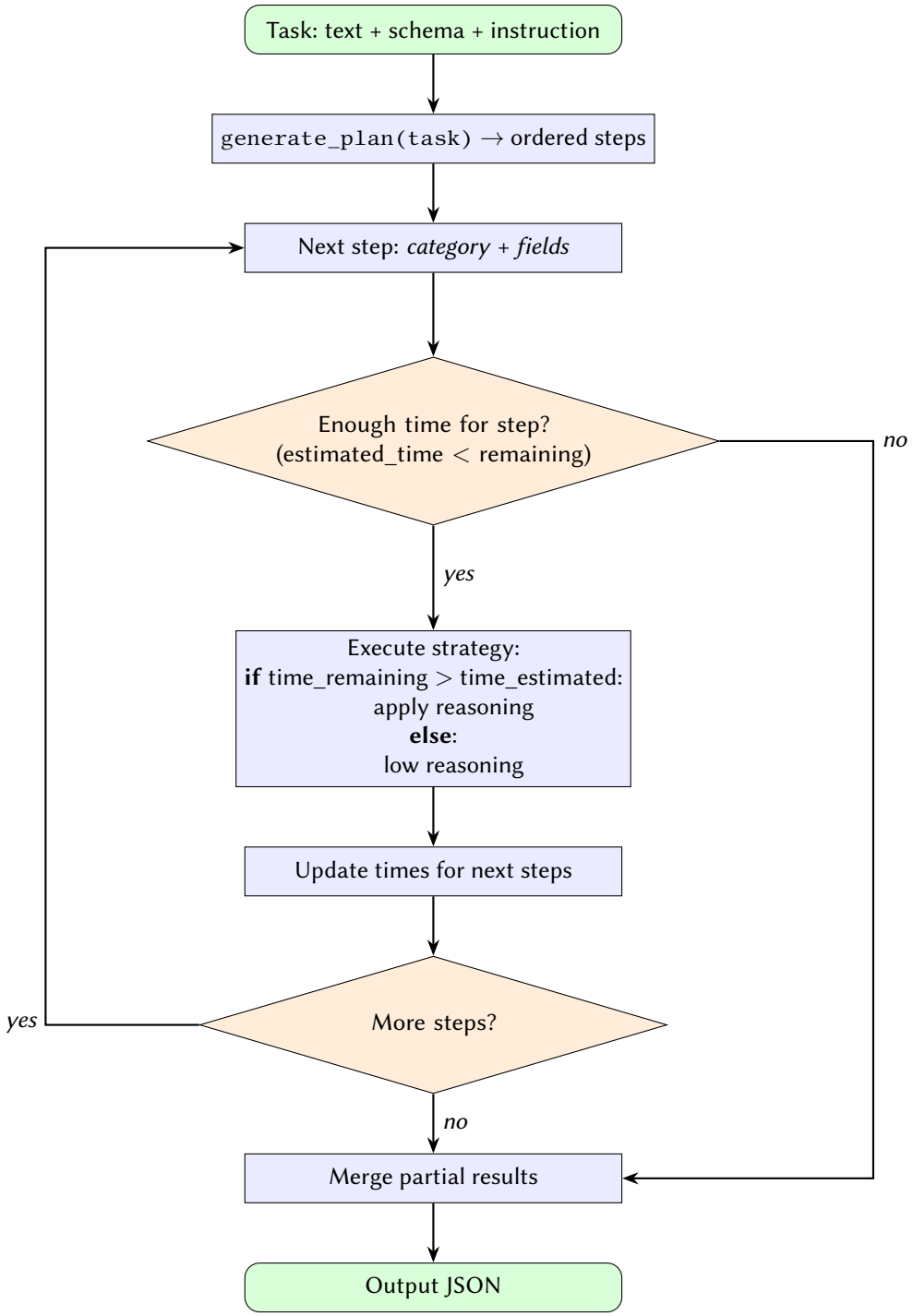


Figure 12: Execution flow of the *experimental* pipeline (`ExperimentalAgent`). First, the plan is generated, and in each step of the loop, the time is evaluated to calculate the estimated time. Also, there is a force out in case the estimated time is longer than the reaming time. Second, the level of reasoning is adjusted with the estimated and remaining time. This process is repeated along all steps in the plan until completion or until all time is consumed. Finally, there is a merge of all partial results. .

Summary. The *limited* pipeline can be described as follows: for each planned extraction step, if (i) the deployed model supports reasoning, (ii) the field category is complex enough to benefit, and (iii) sufficient time remains, a lightweight reasoning pass is added to the extraction call; or a direct call is made. This design allows the same codebase to serve both reasoning and standard models without adding logic in the caller.

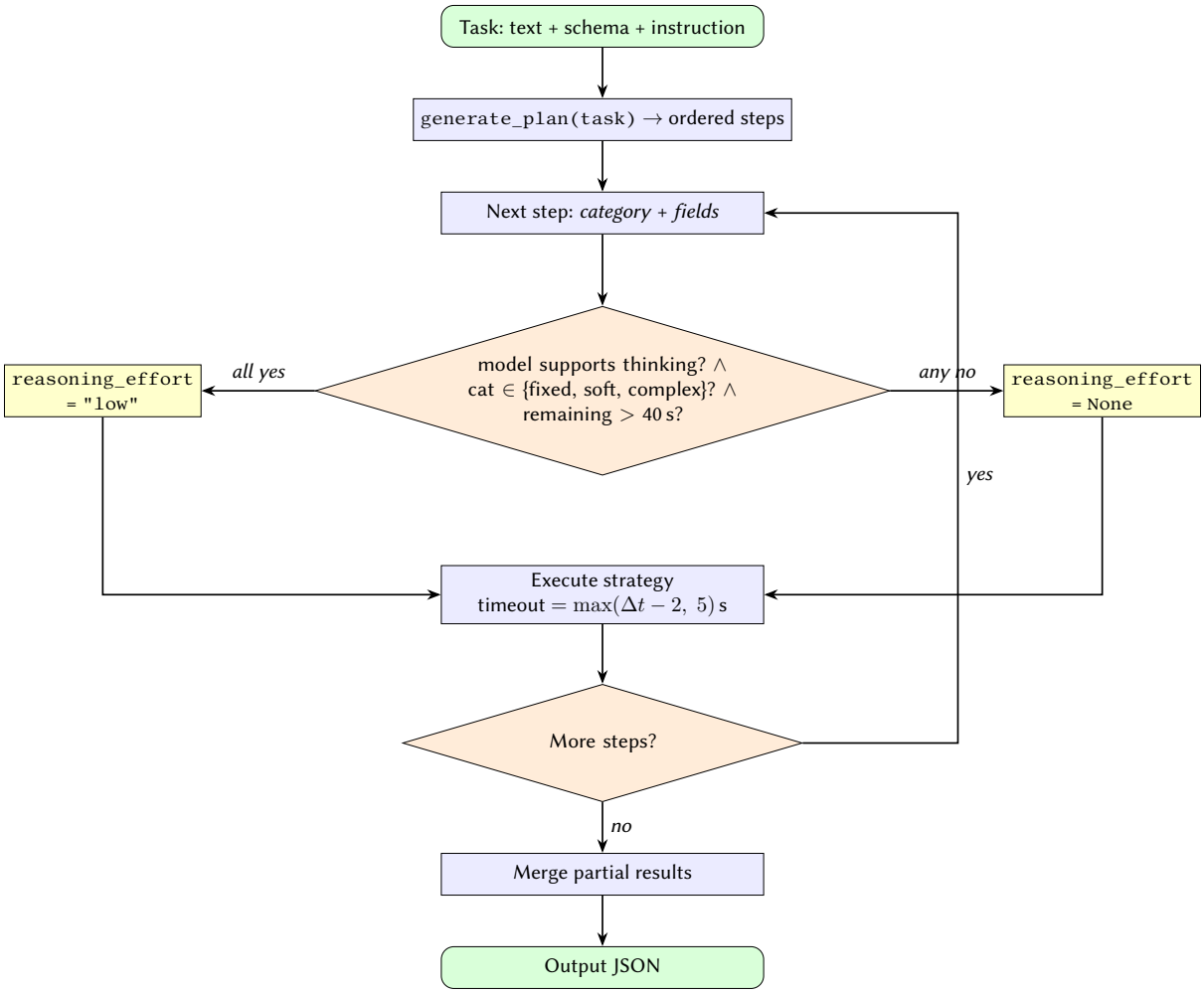


Figure 13: Execution flow of the *limited* pipeline (`ExperimentalAgentV2`). After generating a field-level plan, the pipeline iterates over each step. Lightweight chain-of-thought reasoning (`reasoning_effort="low"`) is activated only when all three conditions hold: the model belongs to a thinking-capable family (Qwen3, QwQ, DeepSeek-R1, O1/O3/O4), the field category is `fixed_entities`, `soft_entities`, or `complex`, and more than 40 s of the 60 s wall-clock budget remain. Each strategy call receives a dynamic timeout that leaves a two-second safety margin.

5. Benchmark and Evaluation

5.1. Experimental setup

To test the adaptability of the different strategies to multiple models we have selected models from different architectures and performance. We tested the Qwen3 models of 1.7B (the smallest model in our test) and 4B [16], the Llama 3.2 model of 3B [17], the Deepseek-r1 of 7B distilled from Qwen [18] and the Gemma4 model of 8B [19]. As a comparison we test the GP4-4o-mini model, a model used commercially that has a good tradeof between computation cost and performance. We compare our strategies against the baseline provided by the organization. The *baseline pipeline*, provided with the code by the organizations of the task, simply performs one call to the API with the schema to obtain the result in one shot. The datasets released before the evaluation phase *starter* and *dev* are used in these experiments. In comparison, the *starter* has less samples (40 vs 149) and less levels of complexity (4 vs 9).

The pipelines are executed using the software provided by the organization. Thus the metrics obtained are directly extracted by the software in its state during GenSIE 2026 evaluation. For these experiments,

the models were dispatched with LMStudio [1].

All the experiments were executed on a local server equipped with two NVIDIA RTX-6000 Ada Generation GPUs (48 GB VRAM each, then 96 GB combined). Due to computational limitations, the experiments were run only once. However, the results might differ substantially among runs due to the stochasticity of the language models and the difficulty of the tasks.

5.2. Evaluation metric

Results are reported using the *Flattened Schema Scoring* metric defined by the task organisers [3]. Both the gold standard and the system output are first flattened into sets of dot-notation key-value pairs (e.g. `event.city="Madrid"`).

Once aligned by key, values are compared field by field according to their schema type:

- **Rigid types** (numbers, booleans, dates, enum strings): binary exact-match score. Case mismatches (e.g. `"positive"` vs. `"POSITIVE"`) score 0, which motivates the enum-normalisation post-processing in our pipelines.
- **Free-text fields** (descriptions, summaries): hybrid score combining cosine similarity of multilingual sentence embeddings (`BAAI/bge-small-en-v1.5`) and normalised lexical overlap, weighted $\alpha = 0.7$ in favour of semantics.
- **Null / hallucination fields**: a hallucinated value where the gold is `null` (or vice-versa) scores 0, directly penalising both over-extraction and under-extraction.
- **Array fields**: order-independent greedy bipartite matching maximises the total similarity before normalising by the Jaccard index of the two lists.

The total *True Positive Score* (TPS) is the sum of per-key similarities across all keys. Micro-Precision, micro-Recall, and micro-F1 are then computed as:

$$\text{Precision} = \frac{\text{TPS}}{|S|}, \quad \text{Recall} = \frac{\text{TPS}}{|G|}, \quad \text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where $|S|$ and $|G|$ are the number of key-value pairs in the system output and the gold standard respectively. The primary leaderboard metric is micro-F1.

6. Results and Analysis

6.1. Results

Table 2 shows the results obtained in the starter dataset. Of the three pipelines that conform the submission, the *stable* is the best performer with all the models. The behaviour of the pipelines are very dissimilar for some models (e.g. *experimental* pipeline did not work well with Qwen3-4B and Deepseek-R1-7B but has competitive results with Gemma4:E4B and Qwen3-17B instead). The *stable* pipeline is the most time consuming due to the use of self-consistency execution during extraction. In fact, the time to reach a solution tends to double or worse the next time consuming pipeline.

The results in the *starter* set are corroborated in the *dev* set (Table 3). Here, the *stable* pipeline did not beat always the other models although it has a competitive performance with all of them. The *stable* pipeline doubles the time of execution of the other pipelines again, despite disabling reasoning in capable models.

One interesting result of these experiments is that Gemm4:e4b gets very close or even beats GPT-4o-mini, a commercially acclaimed model in this task. Qwen3:4B have very similar results to both, showing that model size is not that relevant in this new task. In fact, models such as Deepseek-r1:7B or Llama3.2:3B obtained worse results with all the pipelines to the smaller but more recent Qwen3:1.7B model.

[1] https://lmstudio.ai/

Table 2
Performance results for STARTER dataset.

Dataset	Model	Pipeline	Prec.	Recall	F1	Time
starter	qwen3-1.7b	baseline	0.6881	0.6259	0.6556	36s
starter	**qwen3-1.7b**	**stable**	**0.6537**	**0.6869**	**0.6699**	**7m55s**
starter	qwen3-1.7b	experimental	0.6771	0.4744	0.5579	57s
starter	qwen3-1.7b	limited	0.6293	0.6293	0.6293	4m32s
starter	llama-3.2-3b	baseline	0.7061	0.5904	0.6431	40s
starter	**llama-3.2-3b**	**stable**	**0.6303**	**0.6624**	**0.6460**	**12m0s**
starter	llama-3.2-3b	experimental	0.6432	0.6232	0.6321	2m36s
starter	llama-3.2-3b	limited	0.5428	0.6746	0.6015	1m32s
starter	qwen3-4b	baseline	0.7094	0.7294	0.7192	56s
starter	**qwen3-4b**	**stable**	**0.7241**	**0.7609**	**0.7420**	**11m1s**
starter	qwen3-4b	experimental	0.7107	0.2088	0.3228	28m28s
starter	qwen3-4b	limited	0.6454	0.6736	0.6587	5m19s
starter	deepseek-r1-distill-qwen-7b	baseline	0.6836	0.5060	0.5815	56s
starter	**deepseek-r1-distill-qwen-7b**	**stable**	**0.5948**	**0.6250**	**0.6095**	**17m48s**
starter	deepseek-r1-distill-qwen-7b	experimental	0.5719	0.3360	0.4233	1m59s
starter	deepseek-r1-distill-qwen-7b	limited	0.5181	0.6000	0.5560	6m51s
starter	gemma4:e4b	baseline	0.7689	0.6907	0.7277	4m59s
starter	**gemma4:e4b**	**stable**	**0.7404**	**0.7780**	**0.7587**	**27m40s**
starter	gemma4:e4b	experimental	0.7189	0.7352	0.7270	14m28s
starter	gemma4:e4b	limited	0.7258	0.7627	0.7438	8m45s
starter	gpt-4o-mini	baseline	0.6934	0.5602	0.6197	2m26s
starter	**gpt-4o-mini**	**stable**	**0.7500**	**0.7881**	**0.7686**	**13m40s**
starter	gpt-4o-mini	experimental	0.6618	0.6581	0.6599	6m7s
starter	gpt-4o-mini	limited	0.6201	0.7672	0.6859	4m19s

Table 3
Performance results for DEV dataset.

Dataset	Model	Pipeline	Prec.	Recall	F1	Time
dev	qwen3-1.7b	baseline	0.6672	0.6186	0.6420	5m33s
dev	qwen3-1.7b	stable	0.6701	0.6553	0.6626	48m24s
dev	qwen3-1.7b	experimental	0.6811	0.3433	0.4565	20m23s
dev	**qwen3-1.7b**	**limited**	**0.7221**	**0.6715**	**0.6958**	**34m57s**
dev	llama-3.2-3b	baseline	0.6726	0.6285	0.6498	5m1s
dev	llama-3.2-3b	stable	0.7008	0.6853	0.6930	54m36s
dev	llama-3.2-3b	experimental	0.6911	0.6064	0.6460	23m11s
dev	**llama-3.2-3b**	**limited**	**0.7318**	**0.7325**	**0.7322**	**23m36s**
dev	qwen3-4b	baseline	0.6972	0.6756	0.6862	7m5s
dev	**qwen3-4b**	**stable**	**0.7445**	**0.7445**	**0.7445**	**1h9m45s**
dev	qwen3-4b	experimental	0.7537	0.7116	0.7321	23m04s
dev	qwen3-4b	limited	0.7318	0.7325	0.7322	23m36s
dev	deepseek-r1-distill-qwen-7b	baseline	0.6252	0.5292	0.5732	7m17s
dev	**deepseek-r1-distill-qwen-7b**	**stable**	**0.6554**	**0.6408**	**0.6480**	**1h19m21s**
dev	deepseek-r1-distill-qwen-7b	experimental	0.5827	0.2141	0.3131	17m20s
dev	deepseek-r1-distill-qwen-7b	limited	0.4341	0.6047	0.5054	36m51s
dev	gemma4:e4b	baseline	0.7789	0.7153	0.7457	25m51s
dev	gemma4:e4b	stable	0.7805	0.7521	0.7661	1h57m12s
dev	gemma4:e4b	experimental	0.7839	0.6963	0.7375	1h15m52s
dev	**gemma4:e4b**	**limited**	**0.7951**	**0.7817**	**0.7883**	**54m2s**
dev	gpt-4o-mini	baseline	0.7340	0.5302	0.6156	9m56s
dev	**gpt-4o-mini**	**stable**	**0.7547**	**0.7547**	**0.7547**	**1h15m22s**
dev	gpt-4o-mini	experimental	0.7256	0.6567	0.6895	35m35s
dev	gpt-4o-mini	limited	0.6478	0.7668	0.7023	23m59s

Finally, the results for the test datasets have been plotted in Table 4. This table shows how the complexity and the length of the test datasets have affected both the execution time and the performance of the models. Before starting the analysis, it is important to menthiong that the deepseek model have changed due to problmens in the execution with the model with 7b not related with the model itself but

with the symtem to save the results. It makes no importat difretence but it is expected that the model with 8b and no distilled works better than the distil 7b deepsekk model.

Table 4
Performance results for TEST dataset.

Dataset	Model	Pipeline	Prec.	Recall	F1	Time
test	qwen14b	baseline	0.796	0.578	0.67	0h32m57s
test	**qwen14b**	**stable**	**0.788**	**0.65**	**0.713**	**2h19m42s**
test	qwen14b	experimental	0.763	0.131	0.224	0h41m23s
test	qwen14b	limited	0.69	0.72	0.705	1h32m28s
test	qwen3:1.7b	baseline	0.717	0.594	0.649	40m36s
test	**qwen3:1.7b**	**stable**	**0.686**	**0.623**	**0.653**	**1h36m36s**
test	qwen3:1.7b	experimental	0.665	0.232	0.344	1h26m31s
test	qwen3:1.7b	limited	0.406	0.649	0.499	56m51s
test	llama3.2:3b	baseline	0.637	0.513	0.568	29m15s
test	**llama3.2:3b**	**stable**	**0.658**	**0.565**	**0.608**	**1h47m09s**
test	llama3.2:3b	experimental	0.634	0.254	0.362	40m06s
test	llama3.2:3b	limited	0.28	0.574	0.376	1h29m49
test	**qwen3:4b**	**baseline**	**0.81**	**0.675**	**0.736**	**2h20m51s**
test	qwen3:4b	stable	0.142	0.02	0.035	2h05m22s
test	qwen3:4b	experimental	0.844	0.354	0.499	3h58m48s
test	qwen3:4b	limited	0.916	0.23	0.367	2h36m35s
test	**deepseek-r1:8b**	**baseline**	**0.764**	**0.629**	**0.69**	**1h22m16**
test	deepseek-r1:8b	stable	0.719	0.439	0.545	2h08m07s
test	deepseek-r1:8b	experimental	0.787	0.425	0.552	2h52m47s
test	deepseek-r1:8b	limited	0.719	0.495	0.586	5h33m36s
test	**gemma4:e4b**	**baseline**	**0.7787**	**0.6340**	**0.6989**	**1h28m55s**
test	gemma4:e4b	stable	0.7688	0.5841	0.6639	2h3m41s
test	gemma4:e4b	experimental	0.7403	0.3244	0.4511	1h17m28s
test	gemma4:e4b	limited	0.5670	0.7146	0.6323	1h50m58s
test	qwen3:14b	baseline	0.7962	0.5781	0.6698	33m1s
test	**qwen3:14b**	**stable**	**0.7884**	**0.6500**	**0.7126**	**2h19m45s**
test	qwen3:14b	experimental	0.7632	0.1313	0.2240	41m26s
test	qwen3:14b	limited	0.6904	0.7196	0.7047	1h32m30s

In general, the execution time has been increased. In the extreme case, with the qwen3:4b for the experimental pipeline, the growth has been from 23 minutes to 3 hours and 58 minutes. Also, some strange behaviour has been found: a low F1 in a configuration that previously obtained competitive results. For example, qwen3:1.7b and llama3.2 for experimental and limited, and qwen3:4b for limited and stable pipelines. The case of the qwen3:4b for the stable pipeline is the one that obtained the worst decrease, which has motivated the analysis of the execution that will be done in the next subsection.

Regarding the performance analysis, for the test dataset, the highest performance for F1 is the baseline pipeline, with only significant differences for the qwen3:4b and deepseek models. However, without the execution analysis, with the slight difference in performance, and the high differences in execution time, we cannot extract a strong conclusion from these results.

6.2. Setup and experiment analysis

Two different environments have been set up to run these experiments. The first one is a server with an NVIDIA RTX A6000 GPU and Ollama as an LLM provider. The other one is a server with two NVIDIA RTX A6000 GPUs and LLMStudio as the LLM provider. The first server has run the tests for qwen3-1.7b, llama-3.2-3n, and deepseek-7b and 8b, while the second server has run the qwen3:14b and gemma4 models.

Table 5 shows the exection metrics for the results in table 4. Column "Pass" shows the number of

runs that sucessfully ended, columns "Fail" the ones that failed, and "Timeouts" the ones that ended with timeout. It is importante to understand that the timeout are not counted like fails, as shows the sixth row, the one for qwen3:4b and stable pipeline, where there is no fail but two timeout wich are count as sucsess execution.

Table 5
Pass and fails for test dataset execution.

Model	Pipeline	Pass	Fail	Timeouts	Pass(%)	Fail(%)	Timeout rate(%)	Avg. Time(s)
qwen14b	baseline	127	18	0	87.586	12.414	0.0	13.64
qwen14b	stable	127	18	66	87.586	12.414	45.517	57.808
qwen14b	experimental	67	78	6	46.207	53.793	4.138	17.13
qwen14b	limited	135	10	36	93.103	6.897	24.828	38.266
qwen3:4b	baseline	133	12	67	91.724	8.276	46.207	56.707
qwen3:4b	stable	145	0	2	100.0	0.0	1.379	51.879
qwen3:4b	experimental	106	39	87	73.103	26.897	60.0	98.815
qwen3:4b	limited	135	10	83	93.103	6.897	57.241	64.8
gemma4b	baseline	137	8	10	94.483	5.517	6.897	36.775
gemma4b	stable	145	0	16	100.0	0.0	11.034	51.126
gemma4b	limited	135	10	44	93.103	6.897	30.345	45.903
gemma4b	experimental	76	69	0	52.414	47.586	0.0	0.0
llama3.2:3b	baseline	135	10	1	93.103	6.897	0.69	12.105
llama3.2:3b	stable	135	10	31	93.103	6.897	21.379	44.34
llama3.2:3b	experimental	66	79	6	45.517	54.483	4.138	16.599
llama3.2:3b	limited	135	10	44	93.103	6.897	30.345	37.169
deepseek-r1:8b	baseline	135	10	25	93.103	6.897	17.241	34.048
deepseek-r1:8b	stable	142	3	34	97.931	2.069	23.448	53.018
deepseek-r1:8b	limited	130	15	123	89.655	10.345	84.828	138.044
deepseek-r1:8b	experimental	122	23	74	84.138	15.862	51.034	71.5
qwen3:1.7b	baseline	129	16	4	88.966	11.034	2.759	16.806
qwen3:1.7b	stable	135	10	2	93.103	6.897	1.379	39.979
qwen3:1.7b	experimental	56	89	10	38.621	61.379	6.897	35.804
qwen3:1.7b	limited	135	10	2	93.103	6.897	1.379	23.527

In general, we can observe that the experimental pipeline always fails more than the other pipelines. That means that something in the code, independently of the models, breaks the execution for the experimental pipeline. This is not strange, because the experimental pipeline is the more complex one, and the differences in the format convey the complexity in the test dataset, as well as the complexity of the fields could break the code of the experimental pipeline.

Also, it seems that the stable pipeline gets more successful execution than the baseline pipeline, as is shown for all models but llama3.2:3b, in which both pipelines achieve 135 runs successfully. The limited pipeline also accomplishes a high level of passed executions, but it is lower than the stable pipeline more often.

Analysing the timeout, it seems no consisten pattern which allow us to make some conclusions. For low-parameters qwen models, the experimental pipeline always has the highest tiemout rate, followed by the baseline. However, for the qwen3:4b it is so noticable the diferente of timeouts for the stable pipele: while the others piples have between 10 and 39 timeouts, stable pipeline only threw 2 timeouts. This bevaiout also is repreted for gemma4:4b.

For the experimental and limited pipelines, it is complicated to draw some conclusions regarding the timeout. In general, it seems they are slower than baseline and stable, but not so much if we pay attention to qwen3:4b results. However, for Qwen3:1.7b, it seems that only the experimental pipeline is significantly slower than the others. But if we look at gemma4b, the experimental pipeline has no timeouts, surely because the amount of errors makes it not possible to launch a timeout. Even with that, it also does not consist of the behaviour of the limited pipeline for its timeout rate, because for the qwen3:1.7b model it performs closely similar to baseline and stable, but for the rest of the models it

seems close to the experimental behaviour.

6.3. Complexity and domain analysis

Another interesting analysis is the F_1 metric of different models alongside de domain and the complexity. For this analysis, the *Gemma4:e4b* and *Qwen3:4b* models had been selected because they have performed very competitively in the benchmarks. We tested them using the *stable* pipeline with the idea of obtaining the individual F_1 score in each task, aggregating it by domain, complexity level and data types.

Table 6 shows F_1 score tend to decrease as complexity increases. For the *Gemma4:e4b* model, this decrease is obvious except for $L3$, that failed due to execution errors. In the *Qwen3:4b* model column, the decrease is not as linear as in the other columns because $L3$ increases its F_1, respecting its previous level. Also, $L5$ seems to recover a high F_1 to initiate a decrease -now linear- until $L10$.

Table 6
F1 score of each complexity level for Gemma4:e4b and Qwen3:4b in the *dev* dataset with the *stable* pipeline

Complexity	F1 (gemma)	F1(qwen)
L1	0.92	0.85
L2	0.77	0.71
L3	0	0.76
L5	0.76	0.71
L7	0.81	0.84
L8	0.81	0.79
L9	0.75	0.74
L10	0.64	0.63

For the domain (Table 7), the interesting insight is that models tend to perform very similar, which suggests there is a strong correlation between the scores of both models that are good performers. In other words, easy tasks can be performed proficiently by the small language models whilst difficult tasks are equally challenging.

Table 7
F_1 score for each domain with the models Gemma4:e4b and Qwen3:4b in the *dev* dataset with the *stable* pipeline

Domain	F1 (gemma)	F1(qwen)
stem	0.91	0.91
cultural	0.64	0.75
general	0.79	0.7
legal	0.78	0.72
medical	0.72	0.6
environmental	0.79	0.75
technical	0.82	0.76
lifestyle	0.64	0.63

6.4. Correlation analysis

We have considered it important to study the correlation between two models: *Qwen3:4b* and *Gemma4:e4b*. The Pearson correlation has been chosen to measure the strength and direction of the linear relationship between two series. In this coefficient, the correlation (r) measures the correlation: from 0.7 to 1 is considered a very strong correlation, from 0.4 to 0.7 a moderate correlation, from 0.1 to 0.4 a weak correlation, and from 0 to 0.1 no correlation. Also, it is important to consider the p-value of the metric, because values above 0.05 are statistically insignificant.

Besides, it is also important to present how the experiments have been conducted. For stable and baseline pipelines, different series of data for each model have been chosen. The first series was the F_1 by task, and the second series was the TPS for each field. For the case of the F_1 for tasks, the task name was used to sort the data series for each model, and, in the case of the TPS, the task name and the field name were joined to use them as a key to sort both series of data. That was important to align both series and fill each position of the array with the same source of data.

Then, for each metric (F_1 and TPS), the correlation has been measured following four different criteria to filter the data before correlating the series:

1. By complexity
2. By domain
3. By category/strategy
4. By type

The correlation metric was calculated in the same way for all these criteria: first, the result for one model was filtered for each complexity/domain/category or type, and then, the Pearson correlation was estimated for each subset. The only difference in how the subsets have been split is filtering the types, as one task/field could have more than one type. The subset for each type contains the tasks or fields that have that type between their output types. That means that it could have the same task in two or more series for one type. For example, if one field contains types like numerical or null, it will belong to the subset of numerical type and also to the subset of null type.

After these considerations, Table 8 shows how the meaningful measurements for the correlation appear when using the TPS metric. Most of the results for the R coefficient are not relevant because the p-value for the tasks measures is above 0.05. However, for the TPS metric, it seems that the correlation is from moderate to strong in most cases in baseline and stable pipelines.

Pipeline	**baseline**				**stable**			
Analise	*task*		*fields*		*task*		*fields*	
Complexity	**r**	**p**	**r**	**p**	**r**	**p**	**r**	**p**
L1	0.43	0.25	0.19	0.63	-0.15	0.7	0	1
L2	0.61	0.06	0.83	**0**	0.58	0.08	0.7	**0**
L3								
L4	0.39	**0.01**	0.63	**0**	0.38	**0.02**	0.6	**0**
L5	0.43	**0**	0.49	**0**	0.77	**0**	0.69	**0**
L7	-0.14	0.82	0.45	**0.02**	0.91	**0.03**	0.54	**0**
L8	-0.46	0.54	0.64	**0**	0.77	0.23	0.69	**0**
L9	0.54	0.08	0.77	**0**	0.82	**0**	0.91	**0**
L10	-0.22	0.54	0.03	0.84	0.01	0.98	0.45	**0**

Table 8
Pearson correlation and p-value by complexity

Pipeline	**baseline**				**stable**			
Analise	*task*		*fields*		*task*		*fields*	
Domain	**r**	**p**	**r**	**p**	**r**	**p**	**r**	**p**
all	0.32	**0**	0.54	**0**	0.39	**0**	0.62	0
cultural	-0.24	0.2	0.53	**0**	0.04	0.83	0.59	**0**
environmental	0.32	0.37	0.75	**0**	0.43	0.21	0.67	**0**
general	0.62	0.05	0.68	**0**	0.16	0.65	0.57	**0**
legal	0.56	**0.01**	0.59	**0**	0.67	**0**	0.59	**0**
lifestyle	-0.22	0.54	**0.03**	0.84	0.01	**0.98**	0.45	**0**
medical	0.66	**0**	0.62	**0**	0.42	**0.02**	0.6	**0**
stem	-0.01	0.97	0.22	**0**	0.5	**0.02**	0.43	**0**
technical	0.4	0.11	0.76	**0**	0.55	**0.02**	0.87	**0**

Table 9
Pearson correlation and p-value by domain

In table 9 appears the same behavior as before: most of the measurements for the correlation are not statistically significant as shows the p-value above 0.5 for the taks measuremtns. However, in the TPS metric, it seems that most of the scores have moderate to high correlation in both datasets, except for the stem domain, which always presents the lowest correlation of the statistical mean correlations.

For the study of the correlation by categories, the same categories as in the experimental and limited pipelines have been considered. In this case, we can observe in the Table 10 how the F_1 is also statistically relevant and not only the TPS. The table also shows that the F_1 correlation is weak and moderate for the baseline and moderate for the stable pipeline.

Pipeline	**baseline**				**stable**			
Analise	*task*		*fields*		*task*		*fields*	
Categories	**r**	**p**	**r**	**p**	**r**	**p**	**r**	**p**
direct	0.29	**0**	0.45	**0**	0.38	**0**	0.59	**0**
categorical	0.2	**0.04**	0.55	**0**	0.42	**0**	0.61	**0**
soft_entities	0.54	**0**	0.75	**0**	0.55	**0**	0.7	**0**
fixed_entities	-0.22	0.54	0.49	**0.04**	**0.01**	0.98	0.63	**0**
complex	0.5	**0**	0.48	**0**	0.36	**0.04**	0.37	**0.03**

Table 10
Pearson correlation and p-value by category

Finally, for a fine-grained study of the categories, the table 11 shows the correlation for each type of data. It is important to remember the types explained in 3 to understand the first column of the table. Here, both F_1 and TPS metrics are statistically relevant, probably because one task contains many types of data, and each type contains more tasks. In general, it seems the correlations keep their previous behavior: weak-moderate correlation for F_1 and moderate correlation for TPS.

Pipeline	**baseline**				**stable**			
Analise	*task*		*fields*		*task*		*fields*	
Types	**r**	**p**	**r**	**p**	**r**	**p**	**r**	**p**
$ref	0.2	**0.04**	0.55	**0**	0.42	**0**	0.61	**0**
anyOf	0.29	**0**	0.34	**0**	0.43	**0**	0.55	**0**
array	0.64	**0**	0.72	**0**	0.52	**0**	0.67	**0**
array.$ref	0.5	**0**	0.48	**0**	0.36	**0.04**	0.49	**0**
array.object	0.5	**0**	0.48	**0**	0.36	**0.04**	0.37	**0.03**
boolean	0.1	0.59	0.55	**0**	0.08	0.68	0.19	0.27
integer	0.33	**0**	0.3	**0**	0.37	**0**	0.43	**0**
null	0.29	**0**	0.34	**0**	0.43	**0**	0.55	**0**
number	-0.04	0.8	-0.02	0.83	0.65	**0**	0.23	**0.02**
string	0.29	**0**	0.63	**0**	0.38	**0**	0.62	**0**

Table 11
Pearson correlation and p-value by type

As the reader could have notice, the negative correlations have not been commented. The reason is that any negative correlation was statistically relevant ($p < 0.05$).

7. Discussion

7.1. Observing the Outputs

In traditional machine learning, having a metric for a problem and trying to optimize with a continuous iteration loop is the way to operate. In GenSIE with the different amount of tasks, some of them very subjective, blindly observing the metric could mislead the developments. For this reason, we have developed code to analyze the outputs generated by the system, comparing the expected gold truth with the predictions generated by the pipelines. A markdown organizing this information in tables is

obtained at the end of the process that was manually evaluated. An example of the output can be found in 14. This output help us to understand where the system failed to obtain the expected solution and also to find errors in the gold truth and to provide feedback to the organizers with concrete examples.

cultural_entities_002

Extracts named entity mentions categorized by type in a single list. Complexity: L5 (Hierarchical Grouping).

> Un espectáculo atrevido y extravagante. '¡La novia!' es una fascinante rareza de Hollywood, pero le falta desatarse aún más [...] Cada vez es más raro que en Hollywood confíen grandes presupuestos a películas que se salgan de la norma, pero también es verdad que Warner lleva ya un tiempo sorprendiéndonos. Ya el año pasado llegaron tanto la fallida 'Mickey 17' como la excelente 'Una batalla tras otra', pero se ve que con '¡La novia!' no lo vieron claro y decidieron retrasar su lanzamiento hasta este 6 de marzo de 2026. [...] Mitad secuela apócrifa de 'Frankenstein', mitad reimaginación de 'La novia de Frankenstein', el segundo largometraje como directora de Maggie Gyllenhaal ha contado con 90 millones de presupuesto -inicialmente se dijo que 80, pero Variety ha elevado esa cifra- para dar forma a un espectáculo atrevido y extravagante. El problema es que la actitud excesiva está ahí, pero le falta desatar aún más su locura y eso acaba condenándola a quedarse en una extraña tierra de nadie.

Key	Gold Value	System Value	Score	Match
entities[0↔0].label	PERSON	PERSON	1.0000	✓
entities[0↔0].text	Maggie Gyllenhaal	Maggie Gyllenhaal	1.0000	✓
entities[2↔2].label	ORGANIZATION	ORGANIZATION	1.0000	✓
entities[2↔2].text	Warner	Warner	1.0000	✓
entities[3↔1].label	ORGANIZATION	ORGANIZATION	1.0000	✓
entities[3↔1].text	Hollywood	Hollywood	1.0000	✓
entities[4↔3].label	DATE	DATE	1.0000	✓
entities[4↔3].text	6 de marzo de 2026	6 de marzo de 2026	1.0000	✓
entities[5↔4].label	MISCELLANEOUS	MISCELLANEOUS	1.0000	✓
entities[5↔4].text	¡La novia!	¡La novia!	1.0000	✓
entities[6↔5].label	MISCELLANEOUS	MISCELLANEOUS	1.0000	✓
entities[6↔5].text	Mickey 17	Mickey 17	1.0000	✓
entities[7↔6].label	MISCELLANEOUS	MISCELLANEOUS	1.0000	✓
entities[7↔6].text	Una batalla tras otra	Una batalla tras otra	1.0000	✓
entities[8↔7].label	MISCELLANEOUS	MISCELLANEOUS	1.0000	✓
entities[8↔7].text	Frankenstein	Frankenstein	1.0000	✓
entities[9↔8].label	MISCELLANEOUS	MISCELLANEOUS	1.0000	✓
entities[9↔8].text	La novia de Frankenstein	La novia de Frankenstein	1.0000	✓
entities[1↔9].label	ORGANIZATION	MISCELLANEOUS	0.0000	✗
entities[1↔9].text	Variety	90 millones	0.4143	✗
entities[∅↔10].label	**MISSING**	MISCELLANEOUS	0.0000	⊕
entities[∅↔10].text	**MISSING**	80	0.0000	⊕

TPS: 0.7807 · **P:** 0.7807 · **R:** 0.7807 · **F1:** 0.7807

Figure 14: Evaluation of the task cultural_entities_002 comparing outputs with gold standard

7.2. Dataset Quality and Metrics

The diversity of instructions in GenSIE makes the development of the solutions and the design of the metrics to evaluate them a paramount task. Natural language processing is a field in which many tasks are subjective (e.g. polarity of a content, how similar are two entities) or very difficult to evaluate due to the infinite possible good solutions (e.g. summarization). In the current scenario, all the extractions contribute equally to the final score, disregarding their difficulty. Nonetheless, the schemas are categorized in difficulty levels, thus this information could be used to obtain more insights about how the models perform in each of the tasks.

Not only, building a metric is difficult, but also gathering a corpora large enough and manually validated for this task is very time consuming. We have found errors in the gold truth due to missing entities, or the reverse, entities that were extracted but they do not appear in the text. Some tasks demand the extraction of an unspecified number of elements (e.g. key themes of a movie, ingredients, key people, etc.). Models could generate few or many elements compared to the ground truth. Correcting this behavior might not be easy due to the inconsistency of the test samples due to the subjectivity of the tasks. Figure 15 shows an example of this behaviour in which the pipeline (right column) generates

much more information than the ground truth (left column).

pros[3↔4]	Estilo visual muy personal	un estilo visual muy personal	0.8246	≈
pros[0↔0]	🔴 Sencillez encantadora	una fábula encantadora que te atrapa con su sencillez	0.6557	✗
pros[2↔10]	Banda sonora	destaca la banda sonora del propio director	0.5845	✗
pros[1↔6]	Montaña rusa de emociones	es sencilla y tierna con lecciones necesarias para los más pequeños de la casa	0.4371	✗
pros[∅↔1]	**MISSING**	una encantadora historia en stop-motion nominada al Oscar	0.0000	⊕
pros[∅↔2]	**MISSING**	una de las películas de animación más especiales y emotivas del año	0.0000	⊕
pros[∅↔3]	**MISSING**	una película tierna, que a menudo te deja con una sonrisa y que también te acribilla para sacarte la lágrima	0.0000	⊕
pros[∅↔5]	**MISSING**	con pocos recursos, 'Mi robot favorito' construye la biografía de Celeste a través de los recuerdos de su infancia	0.0000	⊕
pros[∅↔7]	**MISSING**	te remueve el corazón con muchísima nostalgia y punzaditas de amor	0.0000	⊕
pros[∅↔8]	**MISSING**	los sentimientos que transmite son los mismos, aquí que en el espacio exterior	0.0000	⊕
pros[∅↔9]	**MISSING**	🔴 una película encantadora y minimalista	0.0000	⊕

Figure 15: *Pro* arguments of a movie in the *dev* dataset. Left column is the ground truth and the right column is the pipeline execution. Non-verbatim answers are marked with a red dot

Nested structure extractions are also difficult to analyse. If one of the fields of the nested object is wrongly predicted that is used to align the objects (e.g. the tag field), the evaluator might consider all the fields of the nested object as incorrect, penalizing much more the errors in these fields rather than others that are not used to align the objects.

Verbatim extractions can also be problematic. Some instructions demand them, but either gold truth, system predictions or both make subtle changes to the original sentence. Another issue that might arise is what happens with ambiguous instructions, that cannot be fully understood without access to external knowledge. Language models require typically more exhaustive and structured instructions to humans [20, 21] to do the same tasks to mitigate misunderstandings.

7.3. Blackbox models

In GenSIE, models are treated as external entities, which is how commercial models are typically used in practice. Nevertheless, strategies that involve modifications of token sampling or output logit analysis [22] cannot be implemented without accessing to model internals. Thus, despite using open source models, the system interactions are very similar to using third-party APIs. By observing the performance of some of the models such as Gemma4 and Qwen3, it would be interesting to assess how far these small language models could go in solving this task. However, generalization is one of the main motivations of the benchmark and hence, designing a strategy using just one model could be detrimental in the overall performance.

7.4. Methods Discarded

Other methods have been implemented with limited success. Most of them require further analysis to explain why they were ineffective, but due to time constraints, this has not been possible. Nevertheless, we believe it is worth mentioning them below in order to highlight the lines of research on this task that we have ruled out.

Optimisation. An approach to a prompt optimisation system has been implemented. The idea was to implement a system based on LLM calls (one system prompt and one user prompt per call) and to attempt to use models to improve the system for each task. To this end, the inference model and the optimisation model were configured, the former being SLM and the latter a qwen32b. The system was tested with optimisations by task and by task complexity level, yielding very small improvements and, in many cases, with information that was overly overfitted to the task in question. A better optimisation prompt and testing with larger models incorporating reasoning might achieve better optimisation.

Self-optimisation . We also tested a pre-stable system that sought to use the remaining inference time to optimise the system prompt (and the user prompt). The idea was that, after performing an initial inference, the remaining time would be used to ask the model itself to improve the system and user prompts based on the results it had obtained from previous prompts. The result was that it failed to improve them.

RAG. RAG system was tested to retrieve text segments for each question. The idea was to be able to retrieve shorter versions of the input text where the desired answer appeared, a sort of semantic filtering of the input using RAG. It was tested with a miniL2 embedding model, but initial tests showed that while the input was reduced by 50% in terms of characters, the desired answer only appeared in 39% of the fields where RAG was applied. Although progress could have been made in improving the RAG system to try to increase recall (the number of fields in which the desired output did appear), this was discarted due to the limited assistance it provided to the direct inference system using an LLM.

Recurrent Calls. Making multiple calls to the models. The first call was performed in a basic way, with the text and input instructions to provide the output. From then on, as long as there was time, subsequent calls again provided the instructions, the text, and now the output of the previous call as input, indicating whether that output was correct or if it needed any modification or the addition of a field. In the executions, we noticed that with single or text outputs, it didn't present problems, but possibly due to the nature of generating text in the LLMs, in cases where lists had to be generated (listing ingredients, listing steps, listing people, etc.), more lists were generated than were present in the gold data, thus negatively impacting the result.

8. Conclusion

This paper describes three inference pipelines developed for the GenSIE 2026 task on zero-shot structured information extraction using SLMs. The *stable* pipeline demostrates that with the 4 step processing improves over the one-shot baseline, at the cost of significantly higher execution time. The *experimental* pipeline showed that classifying schema fields by semantic type and assigning dedicated strategies per category is a promising direction, though its low recall on some model reveals that the strategy assignment and merging logic requires further refinement maybe caused by using small models. The *limited* pipeline offered the best time and quality tradeoff in most configurations by selectively activating chain-of-thought reasoning only when the time budget and field complexity justified it.

A notable result is that open-weight SLMs, particularly Gemma4:e4b and Qwen3-4B, achieved performance comparable to or better than GPT-4o-mini on this task. This suggests that the complexity of the extraction does not require large models: with careful prompt engineering and multi-step inference design, commodity hardware is sufficient for competitive results.

On the other hand, DeepSeek-R1-Distill-Qwen-7B despite its larger size degraded significantly with more complex pipelines, likely because it always has the reasoning activated, so it consumes a disproportionate amount of the time budget and context window.

Also, we have some observations about the test dataset. The test dataset introduced several characteristics that were not in the other two datasets (starter and dev) and significantly affected the behaviour of our pipelines. The most important difference was the growth in input text length, with a mean character count of 6262 compared to 2412 in dev, and the maximum token lengths reaching approximately 10.000. So this increase penalizes multi-step pipelines because longer inputs translate into longer inference time for the model, and this causes the time budget to be exhausted more frequently. For this reason, we obtain a higher rate of timeouts.

Beyond this length issue, the test dataset introduces shifts into the complexity of the tasks. Level L6, which had never appeared in the previous dataset, was present in the test dataset. The overall complexity distribution became more shifted and concentrated towards high levels, such as L7 and L8.

These changes, combined with the more appearances of anyOf and null fields, indicate that the dataset is not robust, so this leads to failure rates as high as 54% for some model-pipeline combinations.

Talking about the task as a whole, we think that the GenSIE challenge was highly ambitious in scope. It had requiered pipelines to generalize across different SLMs from 1.7B to 14B parameters, without access to model internals and without fine-tuning. This means that no single pipeline can be optimally designed for a specific model's strengths, also taking into account that the capabilities for reasoning of each model are different internally and it also introduce problems in the LLM provider. This makes it difficult to exploit model-specific behaviors that could substantially improve performance.

The execution environment also introduces complexity in the evaluation. Each participant runs inference on their own hardware, differences in GPU memory and LLM serving frameworks, affecting the latency. Also, the LLM providers could influence the results depending on how they handle the memory, the upload and download of the models, and also the bindings for the reasoning capabilities of the model. Although this complexity reflects real-world deployment conditions, it makes it more difficult to discern how the environment around the pipelines influences the metrics.

The increase in complexity and size of the test dataset with respect to the development and starter datasets was also an important concern. Train and test distributions are expected to be sufficiently aligned so that systems optimized on development data can be expected to generalize. In our case, the shift in text length, complexity distribution, and field type proportions between dev and test explains that pipelines that performed competitively during development degraded substantially at evaluation time. Future iterations of the task could benefit from releasing a more representative test set, or alternatively from providing participants with samples or statistics about the expected test distribution so that pipeline design can account for it.

Finally, the task also raised several open challenges that go beyond the design of the pipeline. The evaluation metric penalizes errors in nested structures due to index array alignment and the subjective nature of some extraction targets. We also identified labeling errors in both datasets, which affected our ability to interpret pipeline failures. Future work should explore better alignment strategies for array fields, more robust time estimation for multistep pipelines, and strategies specifically designed to handle the verbatim extraction and hallucination constraints that are central to this task. Finally, evaluating real-world data rather than synthetically generated data would provide a more reliable signal for practical deployment.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to perform grammar and spelling checks and Gemini (3.1 Flash-Lite) for the generation of the schemas of the figures 11, 12 and 13. Authors also used DeepL translator to ensure the correct use of English across different sections. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] R. Grishman, Information extraction: Techniques and challenges, in: International summer school on information extraction, Springer, 1997, pp. 10–27.

[2] 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).

[3] 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.

[4] A. Bonet-Jover, J. A. 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.
[5] H. Gui, S. Qiao, J. Zhang, H. Ye, M. Sun, L. Liang, J. Z. Pan, H. Chen, N. Zhang, Instructie: A bilingual instruction-based information extraction dataset, in: International Semantic Web Conference, Springer, 2024, pp. 59–79.
[6] H. Gui, L. Yuan, H. Ye, N. Zhang, M. Sun, L. Liang, H. Chen, Iepile: Unearthing large-scale schema-based information extraction corpus, arXiv preprint arXiv:2402.14710 (2024).
[7] X. Guan, L. L. Zhang, Y. Liu, N. Shang, Y. Sun, Y. Zhu, F. Yang, M. Yang, Rstar-math: Small llms can master math reasoning with self-evolved deep thinking, arXiv preprint arXiv:2501.04519 (2025).
[8] L. C. Magister, J. Mallinson, J. Adamek, E. Malmi, A. Severyn, Teaching small language models to reason, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2023, pp. 1773–1781.
[9] G. Srivastava, S. Cao, X. Wang, Towards reasoning ability of small language models, arXiv preprint arXiv:2502.11569 (2025).
[10] P. Jhandi, O. Kazi, S. Subramanian, N. Sendas, Small language models for efficient agentic tool calling: Outperforming large models with targeted fine-tuning, arXiv preprint arXiv:2512.15943 (2025).
[11] S. Geng, M. Josifoski, M. Peyrard, R. West, Grammar-constrained decoding for structured nlp tasks without finetuning, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023, pp. 10932–10952.
[12] I. Lee, L. D'Antoni, T. Berg-Kirkpatrick, Good-enough structured generation: A case study on json schema, in: NeurIPS 2025 Fourth Workshop on Deep Learning for Code, ????
[13] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, D. Zhou, Self-consistency improves chain of thought reasoning in language models, arXiv preprint arXiv:2203.11171 (2022).
[14] O. Sainz, I. García-Ferrero, R. Agerri, O. Lacalle, G. Rigau, E. Agirre, Gollie: Annotation guidelines improve zero-shot information-extraction, in: International Conference on Learning Representations, volume 2024, 2024, pp. 47083–47107.
[15] S. Liang, Y. Zhang, Y. Wu, R. Tang, Y. Liu, Schema as parameterized tools for universal information extraction, arXiv preprint arXiv:2506.01276 (2025).
[16] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al., Qwen3 technical report, arXiv preprint arXiv:2505.09388 (2025).
[17] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al., The llama 3 herd of models, arXiv preprint arXiv:2407.21783 (2024).
[18] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, et al., Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, arXiv preprint arXiv:2501.12948 (2025).
[19] M. M. H. Manik, G. Wang, Unified deployment-aware evaluation of open reasoning language models, arXiv preprint arXiv:2604.07035 (2026).
[20] Z. Lin, How to write effective prompts for large language models, Nature Human Behaviour 8 (2024) 611–615.
[21] A. Gao, Prompt engineering for large language models, Available at SSRN 4504303 (2023).
[22] Y. Chang, B. Cao, L. Lin, Monitoring decoding: Mitigating hallucination via evaluating the factuality of partial response during generation, in: Findings of the Association for Computational Linguistics: ACL 2025, 2025, pp. 14574–14587.