UniTor at Cruciverb-IT: Retrieval-Augmented Two-Step Reasoning for Italian Crossword Clue Answering

Andriy Shcherbakov[1], Danilo Croce[2] and Roberto Basili[2]

[1]*Reveal S.R.L., via Kenia 21, 00144, Rome, Italy*
[2]*Department of Enterprise Engineering, University of Rome Tor Vergata*
Via del Politecnico 1, 00133, Rome, Italy

Abstract

Crossword clue answering requires the integration of lexical knowledge, semantic reasoning, and strict structural constraints. We present **UniTor**, an LLM-based system for Italian crossword clue answering, submitted to the Cruciverb-IT Task 1 at EVALITA 2026. UniTor formulates clue solving as a constrained ranking problem and combines retrieval-augmented evidence with a structured two-step prompting strategy executed within a single model call. In the first step, the model generates a diverse set of plausible candidate answers; in the second step, it enforces length constraints and re-ranks the candidates using explicit confidence scores. Extensive ablation experiments demonstrate that retrieval-based grounding and two-step reasoning provide complementary benefits, improving both recall and ranking stability across models of different sizes. Without any task-specific fine-tuning, UniTor achieves the highest score in the official evaluation.

Keywords

Crossword Solving, Large Language Models, Information Retrieval, Reasoning, Italian NLP

1. Introduction

Crossword clue answering poses a challenging problem for Natural Language Processing, as it requires mapping short, often indirect textual clues to correct solutions under strict structural constraints. Unlike standard question answering, crossword clues rarely correspond to explicit definitions and frequently exploit polysemy, figurative language, abbreviations, cultural references, and domain-specific conventions. Moreover, candidate solutions must satisfy hard constraints such as an exact character length and, in grid-based settings, cross-letter consistency.

The Cruciverb-IT shared task [1] at EVALITA 2026 [2] formalizes this problem as a ranking task: given an Italian clue c and a target answer length ℓ, systems are required to output a list of candidate solutions ordered by probability. This formulation reflects realistic crossword-solving conditions, where multiple candidates may be linguistically plausible and the correct solution is often determined only by additional constraints. For instance, for the clue "*Un quadro composto da frammenti*"[1], both *Collage* and *Mosaico* are valid semantic interpretations, and disambiguation cannot rely on meaning alone. As a result, crossword solving is more naturally modeled as a constrained ranking problem rather than as single-label prediction.

Previous work on Italian crosswords has shown that generic lexical or semantic similarity methods are insufficient to capture the complex relationship between clues and solutions. In particular, [3] frames clue answering as a specialized information retrieval problem, highlighting how standard matching approaches struggle when clues involve non-literal language or crossword-specific reasoning. These observations motivate approaches that combine high-recall retrieval with more structured inference and selection mechanisms.

Large Language Models (LLMs) constitute a promising foundation for this task, as they encode extensive lexical knowledge and exhibit strong associative and compositional reasoning abilities. Scaling-

EVALITA 2026: 9th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, Feb 26 - 27, Bari, IT
shcherbakov@revealsrl.it (A. Shcherbakov); croce@info.uniroma2.it (D. Croce); basili@info.uniroma2.it (R. Basili)
0000-0001-9111-1950 (D. Croce); 0000-0001-5140-0694 (R. Basili)

[1]Italian clue translation: "*A picture composed of fragments.*" The candidate answers are *Collage*, translated as "*collage*," and Mosaico, translated as "*mosaic.*"

law studies suggest that increasing model size improves general language competence and knowledge coverage [4, 5]. However, when applied naively to constrained generation tasks, LLMs often exhibit undesirable behaviors, including violations of hard constraints, hallucination of rare or invalid forms, and unstable rankings across runs. These issues are particularly critical in CRUCIVERB-IT, where systems are evaluated on top-k accuracy and ranking quality (MRR) [1].

Two complementary directions have been proposed to mitigate these limitations. First, retrieval-augmented generation (RAG) grounds model outputs in external evidence, reducing reliance on parametric memory and improving robustness [6]. Second, structured prompting strategies, such as chain-of-thought and self-consistency, improve reasoning reliability by decomposing inference into multiple stages or separating generation from selection [7, 8]. Recent work further suggests that eliciting explicit probability distributions or diversified outputs can mitigate mode collapse and improve ranking consistency [9].

In this paper, we present **UniTor**, our submission to the CRUCIVERB-IT Task 1 [1]. UniTor is an LLM-based crossword clue solver built around three key design principles. First, it leverages retrieval-augmented suggestions obtained from a large repository of clue–solution pairs, using a length-constrained retrieval pipeline over a large clue–solution repository (we evaluated lexical, neural, and hybrid variants; the final configuration uses neural retrieval). Second, it adopts a two-step reasoning strategy that explicitly separates candidate exploration from constraint-aware re-ranking within a single LLM call. Third, it requires the model to output explicit probability scores for each candidate, aiming to stabilize rankings and support controlled diversity.

Beyond introducing the system, we use the CRUCIVERB-IT task as a controlled experimental setting to investigate four practical research questions: (i) whether requesting probabilistic outputs improves ranking quality over simple list generation; (ii) whether two-step prompting outperforms single-pass inference; (iii) whether and to what extent model scale affects performance; and (iv) how retrieval-based evidence interacts with model scale and prompting strategies. Our final system achieves the best performance in the official evaluation, ranking first among all submitted systems.

In the remainder of the paper, we review related work (Section 2), describe the UniTor system in detail (Section 3), present experimental results and ablation studies (Section 4), and conclude with a discussion of future directions.

2. Related Work

Early approaches to automatic crossword solving predominantly relied on lexical resources, handcrafted features, and traditional information retrieval techniques — such as BM25, edit distance, and surface-level similarity measures — to match clues with candidate answers. Systems in this category typically leveraged large clue–answer databases and heuristic ranking strategies, achieving reasonable performance on straightforward definitions but exhibiting limited robustness when confronted with indirect formulations, wordplay, or culturally grounded clues.

In the English setting, canonical examples of this paradigm include the Proverb system [10], which combined a large crossword database with expert modules and constraint satisfaction techniques to solve American-style crosswords. Similarly, WebCrow [11] explored the use of the web as a primary knowledge source, reducing dependence on handcrafted knowledge modules while still relying on heuristic retrieval and ranking. Closely related work on language games framed clue answering as a knowledge-based association problem and exploited large cultural repositories to identify candidates [12]. In the Italian crossword setting, Barlacchi et al. [13] and Moschitti et al. [14] study clue answering as a retrieval-and-ranking problem, showing that learning-to-rank and syntax-aware ranking strategies can improve ordering quality over pure retrieval baselines.

More recent work has shifted toward neural representations and learned retrieval models. In particular, Crossword Space [3] models clues and solutions as distinct "modalities" embedded in a shared latent space, training dual-encoder architectures with contrastive objectives on large Italian crossword datasets. These approaches outperform generic encoders and traditional retrieval baselines, highlighting

the importance of domain-specific representations for capturing polysemy, lateral associations, and crossword-specific conventions, and showing promising generalization to previously unseen words.

In parallel, Large Language Models (LLMs) have recently been investigated as solvers for crossword puzzles and related constrained language tasks [15]. Although LLMs demonstrate strong capabilities in paraphrasing and associative inference, their direct application to crossword solving remains challenging due to violations of hard structural constraints such as exact answer length. For instance, Saha et al. [15] report systematic difficulties in enforcing character-count constraints, especially for rare targets. The cryptic crossword setting is even more challenging, as it requires wordplay-driven inference that current LLMs still handle unreliably [16].

Several strategies have been proposed to mitigate these limitations. Retrieval-Augmented Generation (RAG) grounds model outputs in external evidence, improving robustness and selection quality [6]. Prompting strategies that elicit explicit reasoning or decompose inference into stages (e.g., chain-of-thought and related approaches) have been shown to enhance reliability in constrained tasks [7, 8]. More broadly, techniques that encourage diversified outputs and reduce mode collapse can further support stable ranking under ambiguity [9]. However, comparatively little work has systematically analyzed the interaction between retrieval-based grounding, prompt structure, and probabilistic ranking in the specific context of crossword clue answering for non-English languages such as Italian.

Finally, recent shared-task settings emphasize crossword clue answering as a constrained ranking problem. In particular, CRUCIVERB-IT at EVALITA 2026 evaluates systems that must return probability-ordered candidate lists under exact length constraints, using metrics such as Accuracy@k and MRR [1]. This formulation encourages approaches that combine high-recall retrieval with robust ranking and constraint handling.

Our work builds on these contributions by integrating retrieval-based suggestion generation, multi-step prompting, and probabilistic ranking within a unified LLM-based framework tailored to CRUCIVERB-IT. Unlike approaches that rely solely on learned retrieval models or unconstrained LLM generation, our system uses retrieval as evidence while delegating fine-grained discrimination among close alternatives to an LLM-based re-ranking step, enabling controlled ablations that quantify the impact of individual components.

3. System Description

In CRUCIVERB-IT Task 1, each input instance is defined as a pair (c, ℓ), where c is an Italian crossword clue and $\ell \in \mathbb{N}$ denotes the exact character length of the target solution. Given (c, ℓ), a system is required to produce an ordered list of candidate solutions $S = \langle s_1, \ldots, s_K \rangle$ together with associated scores $P = \langle p_1, \ldots, p_K \rangle$, such that $p_i \in [0, 1]$ and $p_i \geq p_{i+1}\ \forall i \in [1, K-1]$ [1].

The scores are interpreted as confidence estimates rather than as a probability distribution: they are not required to sum to one and are used exclusively for ranking purposes. This formulation explicitly models crossword solving as a ranking problem, in which multiple candidates may be linguistically plausible and the correct solution is expected to appear among the highest-ranked entries.

Normalization and structural validity. To enforce crossword-specific constraints, we define a normalization operator $\mathcal{N}(\cdot)$ that maps a string to a canonical form. *A string is considered canonical if it is fully uppercased and contains only Latin uppercase characters in the range A–Z.* Accordingly, normalization consists of converting the input to uppercase and removing all characters outside this range, including accents, apostrophes, spaces, hyphens, and other non-alphabetic symbols.

A candidate solution s is said to be *structurally valid* with respect to a target length ℓ if:

$$\text{valid}(s, \ell) \iff |\mathcal{N}(s)| = \ell$$

When a pattern (or schema) is provided, structural validity additionally requires that all fixed letters in the schema match the corresponding positions in $\mathcal{N}(s)$. In the CRUCIVERB-IT Task 1 setting considered in

this work, no schema constraint is provided; therefore, we enforce only normalization and exact-length constraints.

Example. For the input $c =$ "Capitale della Norvegia"[2] and $\ell = 4$, a correct system output may begin with:

$$\langle (\textsc{oslo}, 0.92), (\textsc{moss}, 0.21), (\textsc{alta}, 0.15), \dots \rangle.$$

Candidates such as OSLO_ or BODØ are mapped by $\mathcal{N}(\cdot)$ to OSLO and BODO, respectively, and are therefore *structurally valid*. In contrast, ASKER is *not structurally valid* because $|\mathcal{N}(\textsc{asker})| = 5 \neq 4$.

3.1. Overview of UniTor

UniTor is a retrieval-grounded, LLM-based solver that maps an input pair (c, ℓ) to a probability-ranked list of candidate solutions. The system combines two tightly integrated components: (i) a *suggestion stage*, which retrieves length-compatible evidence from a large repository of clue–solution pairs, and (ii) a *reasoning-and-ranking stage*, in which an LLM generates, filters, and orders candidate solutions conditioned on the retrieved evidence.

The retrieved suggestions act as lightweight supervision in a few-shot-like manner, grounding the model's generation process while preserving flexibility in candidate exploration.

Single-call, two-phase prompting. Although UniTor is conceptually described as a two-stage pipeline, the *reasoning-and-ranking stage is executed within a single LLM call*. Both candidate exploration and final selection are implemented through a single structured prompt that explicitly enforces a two-phase behavior, as detailed in Appendix A.

Concretely, the model is instructed to produce two ordered lists in sequence: (i) an exploratory list of *initial candidates*, designed to maximize recall and diversity and allowed to include candidates that do not satisfy the target length constraint, and (ii) a refined list of *final candidates*, obtained by filtering and re-ranking the initial candidates under hard structural constraints (exact length) and by assigning explicit probability scores.

This design ensures that exploratory reasoning and constraint-aware selection are both exposed and controlled, while avoiding multiple round-trips to the LLM.

Design objectives. The overall architecture of UniTor is driven by three complementary objectives: (a) *high recall* of plausible crossword solutions, achieved through retrieval-augmented suggestions; (b) *ranking quality and stability*, obtained by explicitly separating exploration from selection within the prompt; and (c) *constraint adherence*, enforced through normalization rules and optional post-processing.

3.2. Retrieval-Augmented Suggestions as Few-shot Evidence

UniTor relies on a large indexed repository $\mathcal{D} = \{(d_i, w_i, \ell_i, v_i)\}_{i=1}^{N}$ of approximately $N \approx 375{,}000$ word–definition pairs, where d_i denotes a definition or crossword clue, w_i the corresponding solution word, $\ell_i = |\mathcal{N}(w_i)|$ its normalized length, and v_i a dense vector representation computed using the BGE-M3 embedding model [17].

At inference time, given an input instance (c, ℓ), the search space is first restricted to length-compatible entries:

$$\mathcal{D}_\ell = \{(d_i, w_i, \ell_i, v_i) \in \mathcal{D} \ : \ \ell_i = \ell\}.$$

From this subset, the system retrieves a small set of relevant examples $\mathcal{R}(c, \ell)$.

Crucially, retrieved items are not treated as final answers, but as *few-shot evidence* injected into the LLM prompt. Each retrieved pair provides lightweight in-context supervision of the form *definition* $\rightarrow$ *solution*, biasing the model toward crossword-consistent lexical choices while preserving generative flexibility.

[2] "Capital of Norway"

Retrieval strategies. We experiment with three retrieval strategies over $\mathcal{D}_\ell$:

- **Lexical (BM25):** candidates are ranked according to BM25 similarity between the input clue c and the stored definition d_i [18].
- **Neural:** the clue is encoded as $v_c = E(c)$, where $E(\cdot)$ denotes the BGE-M3 encoder, and candidates are ranked by cosine similarity $\cos(v_c, v_i)$.
- **Hybrid:** the top-M candidates retrieved with BM25 are re-ranked using a combined score, e.g., $\mathrm{BM25}(c, d_i) \cdot \cos(v_c, v_i)$.

Example. For the clue $c =$ "Un quadro composto da frammenti" [3] and target length $\ell = 7$, the retrieval stage may return suggestions such as[4]:

$$\mathcal{R}(c, 7) = \{\ (\textsc{mosaico},\ \textit{Composizione ottenuta assemblando tessere o frammenti}),\\ (\textsc{collage},\ \textit{Quadro realizzato incollando ritagli di carta o materiali diversi}),\ \ldots\ \}$$

These examples are injected into the prompt as contextual evidence. The LLM is then asked to reason over them, disambiguate between semantically related candidates, and produce a probability-ranked output consistent with crossword conventions.

3.3. LLMs and Model Scale

We evaluate UniTor using a set of instruction-tuned Large Language Models (LLMs) that differ substantially in parameter scale, in order to assess whether performance gains in crossword clue answering are primarily driven by raw model capacity, retrieval grounding, or prompting and reasoning strategies.

This comparison is motivated by well-established scaling laws for neural language models, which suggest that larger models tend to encode richer lexical and world knowledge and exhibit stronger in-context learning capabilities. At the same time, crossword solving is a highly constrained task, in which structural validity and ranking stability may benefit more from explicit guidance than from sheer model size.

Concretely, we evaluate the following model families:

- **Llama 3.1 8B Instruct** [19], representing a relatively small but efficient open-weight model.
- **GPT-OSS 20B** and **GPT-OSS 120B** [20], allowing direct comparison between two scales within the same architectural and training paradigm.
- **GLM 4.6** [21], a very large instruction-tuned model with strong reported reasoning capabilities.

By evaluating the same retrieval-augmented, two-step prompting strategy across these models, we aim to disentangle the contribution of model scale from that of evidence injection and structured reasoning. In particular, this setup allows us to test whether smaller models can approach the performance of much larger ones when coupled with retrieval-based few-shot evidence and stabilized through explicit re-ranking.

3.4. Two-Step LLM Reasoning and Ranking

Rather than prompting the LLM to directly output a final ranked list, UniTor enforces a two-step reasoning process *within a single model request.* This design follows the general intuition of structured reasoning prompts and decomposition-based inference [7], while avoiding multiple interactions with the model.

The two steps are explicitly encoded in the prompt and reflected in the structured output, making the intermediate reasoning observable and controllable.

[3] "A picture composed of fragments."

[4] (MOSAIC, composition made by assembling tiles or fragments), (COLLAGE, picture made by pasting paper cut-outs or different materials)

Step 1: Candidate proposal (exploration). In the first phase, the LLM receives the input pair (c, ℓ) together with the retrieved set $\mathcal{R}(c, \ell)$, which is injected as few-shot evidence. The model is instructed to generate up to K *plausible* candidate solutions, prioritizing semantic coverage and diversity rather than strict constraint satisfaction.

This exploratory step encourages the inclusion of: (i) synonyms and near-synonyms, (ii) hypernyms and related lexical items, and (iii) crossword-specific variants (e.g., preferred lemmas or crosswordese). Candidates proposed at this stage may violate the target length constraint and are assigned preliminary confidence scores.

Step 2: Re-ranking under constraints (selection). In the second phase, the LLM is instructed to re-evaluate the candidate set produced in Step 1 and to perform explicit selection and ranking. Concretely, the model is asked to: (i) apply normalization and enforce exact length constraints, (ii) compare candidates explicitly with respect to clue semantics and crossword conventions, and (iii) output a final list of candidates ordered by decreasing probability.

This separation between exploration and selection is intended to improve ranking stability and reduce the impact of spurious or overly salient candidates, in line with prior observations on robustness through selection and re-ranking in LLM reasoning.

3.5. Probabilistic Ranking, Constraints, and Post-processing

The final system output is a JSON-formatted list of pairs (s_i, p_i), where s_i is a candidate solution and $p_i \in [0, 1]$ is an associated score, with candidates sorted in descending order of p_i. The scores are interpreted as *ranking confidences* rather than as a calibrated probability distribution, and therefore are not constrained to sum to one.

Explicit probability assignment serves two purposes: it aligns with the task specification, and it enables more stable ranking behavior by encouraging the model to distribute confidence across multiple plausible solutions rather than collapsing onto a single mode, as suggested by previous work on self-consistency and diversity in LLM outputs [9].

Constraint handling. In UniTor, we consider two constraint-handling configurations:

- **Intrinsic (LLM-only) constraint adherence**: constraints on normalization and length are enforced solely through the prompt, without external filtering. This setting allows us to directly measure the model's ability to internalize and respect hard structural constraints.
- **Post-processed constraint enforcement**: the normalization operator $\mathcal{N}$ is applied programmatically, and any candidate violating $\mathrm{valid}(s, \ell)$ is discarded. If fewer than K candidates remain, additional length-valid candidates may be backfilled from the retrieval set $\mathcal{R}(c, \ell)$.

Post-processing can improve the task scores, and it partially shifts the burden from generation to retrieval. In Section 4 we report results for the first configuration in order to disentangle intrinsic reasoning capabilities from external constraint enforcement.

Summary. UniTor formulates crossword clue answering as a *constrained ranking problem* rather than as single-label prediction. The system combines (i) length-filtered retrieval to provide retrieval-augmented few-shot evidence, (ii) a structured two-step prompting strategy implemented within a single LLM call to separate exploration from selection and improve ranking stability, and (iii) explicit probabilistic outputs to support robust ordering of multiple plausible candidates. This design directly matches the requirements of the CRUCIVERB-IT task [1] and allows us to systematically analyze the impact of retrieval grounding, reasoning decomposition, and probabilistic ranking on crossword clue answering performance. Appendix A provides the full UniTor prompt and the exact output format used in our experiments.

4. Experimental Results

The effectiveness of UniTor has been evaluated through a series of controlled ablation tests whose results are derived from the official CRUCIVERB-IT Task 1 test set. Our experiments are designed to quantify the individual and combined impact of four key design choices: (i) explicit probabilistic outputs, (ii) two-step reasoning within a single LLM call, (iii) retrieval-augmented suggestions, and (iv) model scale.

Experimental setting. UniTor does not rely on any form of task-specific fine-tuning. All experiments are conducted using off-the-shelf, instruction-tuned large language models accessed via prompt-based inference (through the `ollama` framework[5]), without updating model parameters or performing supervised adaptation. This design choice allows us to focus the analysis on the contribution of individual system components—prompt structure, retrieval grounding, and ranking strategy—rather than on training effects.

For ablation experiments, we use the official CRUCIVERB-IT dataset[6] released for EVALITA 2026. Specifically, we randomly sample 1,000 instances from the test split and use them as a validation set for controlled comparisons across different prompt configurations and model choices.

The full training portion of the dataset (approximately 375,000 word–definition pairs) is not used for any model training. Instead, it is employed during a preparatory phase to build the retrieval index used for suggestion generation. The index is implemented using the Qdrant vector database[7] and stores, for each entry, the clue text (definition), the corresponding solution (word) along with its character length. Each clue is further enhanced with a dense representation computed using the BGE-M3 embedding model [17], enabling neural and hybrid retrieval.

Retrieval variants and selection criterion. We experimented with three retrieval strategies: (i) a purely lexical retriever based on BM25, (ii) a neural retriever based on dense embeddings (cosine similarity in the BGE-M3 space), and (iii) a hybrid strategy that combines the two signals by multiplying their scores.[8] Importantly, retrieval quality was not evaluated as a standalone IR task. Instead, we compared retrieval strategies within the full end-to-end pipeline, measuring their impact on the final crossword-solving metrics (Accuracy@1, Accuracy@10 and MRR).

In these experiments, the neural retriever yielded the strongest downstream performance. For space reasons, we report only the neural-retrieval configuration in the main paper, while the BM25 and hybrid variants are omitted.

For prompt configurations requiring 10, 20, or 50 suggestions, we retrieve candidates by enforcing the answer length as a hard constraint and inject the resulting clue–solution examples as contextual evidence. For each prompt configuration, this procedure produces 1,000 instantiated prompts, which are then submitted to the corresponding LLM for inference.

Output validation and formatting issues. After inference, we parse model outputs to extract the candidate lists and discard any extraneous text outside the prescribed format. When a specific output format is required, a small fraction of generations may be malformed (e.g., due to missing brackets, quotes, or comma separators, or because JSON-like content is mixed with comments). In these cases, we apply minimal, deterministic syntactic repairs to recover a valid serialization (e.g., balancing brackets or quotes, or removing inline comments), without altering the set or order of predicted candidate strings.

The frequency of malformed outputs decreases with increasing model size: GLM 4.6 rarely required intervention; GPT-OSS 20B produced approximately 15–20 invalid outputs out of 1,000 (1.5–2%); Llama 3.1 8B produced approximately 25% invalid outputs and was therefore excluded from the detailed analysis.

[5] https://ollama.com/
[6] https://huggingface.co/datasets/cruciverb-it/evalita2026
[7] https://qdrant.tech/
[8] In all cases, answer length is enforced as a hard constraint at retrieval time.

Results. The CRUCIVERB-IT Task 1 evaluates systems in a *constrained ranking* setting: given a clue and a target answer length, systems must output an ordered list of candidates. Performance is measured using ranking-aware metrics, namely Accuracy@k (whether the gold solution appears within the top-k ranked candidates, with $k \in \{1, 10\}$) and MRR (Mean Reciprocal Rank), which accounts for the position of the first correct candidate in the ranked list (higher is better) [1].

Table 1 reports an ablation study conducted without retrieval-augmented suggestions, isolating the effect of prompt design choices and model scale. In particular, we compare: (i) the use of explicit probabilistic outputs versus greedy list generation, and (ii) a single-step versus a two-step reasoning strategy, across three LLMs of increasing size.

Prob.	Two-step	Accuracy@1			Accuracy@10			MRR		
		GPT-OSS 20B	GPT-OSS 120B	GLM 4.6	GPT-OSS 20B	GPT-OSS 120B	GLM 4.6	GPT-OSS 20B	GPT-OSS 120B	GLM 4.6
No	**No**	26.60	45.10	44.60	35.60	55.30	61.50	30.02	49.28	51.09
	Yes	25.80	44.30	50.00	29.50	49.80	60.70	27.36	46.81	54.56
Yes	**No**	24.80	45.50	48.00	32.30	55.30	62.40	27.85	49.67	54.16
	Yes	23.40	44.10	53.80	27.40	50.00	64.40	24.98	46.64	58.29

Table 1
Performance (%) of UniTor variants *without retrieval-augmented suggestions*, analyzing the effect of probabilistic outputs and two-step prompting across different LLMs.

Several trends emerge from the results. First, model scale has a strong and consistent impact across all configurations: larger models substantially outperform smaller ones, confirming that crossword clue answering benefits from increased parametric knowledge and reasoning capacity. However, scale alone is not sufficient to guarantee stable or optimal ranking quality.

Second, the introduction of a two-step reasoning strategy yields clear improvements in ranking-oriented metrics, particularly MRR, for the largest model (GLM 4.6). This suggests that explicitly separating candidate generation from selection helps mitigate ranking instability and overconfident early choices, even in the absence of external evidence.

Third, explicit probabilistic outputs have a nuanced effect. While they do not consistently improve Accuracy@1 in isolation, they tend to benefit ranking quality (MRR and Accuracy@10), especially when combined with two-step prompting. This supports the hypothesis that requiring the model to distribute confidence across multiple candidates reduces mode collapse and improves relative ordering among plausible solutions.

Finally, interactions between prompt structure and model scale are non-trivial. For smaller and mid-sized models, additional structure can sometimes slightly reduce Accuracy@1, likely due to increased cognitive load. In contrast, larger models are able to exploit structured prompts more effectively, translating them into measurable gains in ranking quality.

Table 2 reports results obtained by augmenting the LLM with retrieval-based suggestions, varying both the number of retrieved examples and the use of single-step versus two-step prompting. Following the trends observed in Table 1, we always use the probabilistic output format in this set of experiments, as it provided the most consistent gains in ranking quality—especially for the largest model—while keeping the comparison focused on the effect of retrieval and prompt structure.

In this setting, retrieved clue–solution pairs are injected into the prompt as few-shot evidence, allowing us to assess how external grounding interacts with reasoning structure and model scale.

The results show that retrieval-augmented suggestions have a large and consistent impact on performance across all models and metrics. Even a small number of retrieved examples (10) leads to substantial gains over the no-retrieval baseline, with Accuracy@1 improvements exceeding 10 percentage points for all models.

Increasing the number of retrieved suggestions generally improves performance, but with diminishing returns. Gains between 10 and 20 examples are modest, and in some cases performance plateaus or

Strategy		Accuracy@1			Accuracy@10			MRR		
Two-step	Added Ex.	GPT-OSS 20B	GPT-OSS 120B	GLM 4.6	GPT-OSS 20B	GPT-OSS 120B	GLM 4.6	GPT-OSS 20B	GPT-OSS 120B	GLM 4.6
No	0	24.80	45.50	48.00	32.30	55.30	62.40	27.85	49.67	54.16
	10	51.50	62.80	63.30	64.50	76.10	76.30	56.65	68.15	68.57
	20	51.60	62.00	63.80	65.80	77.60	74.50	57.00	68.22	68.24
	50	50.80	65.60	66.20	63.90	79.90	76.80	55.95	71.26	70.52
Yes	0	24.40	43.80	53.80	28.60	51.30	63.70	26.14	47.03	58.01
	10	47.10	59.40	65.50	58.10	71.40	75.20	51.60	64.46	69.68
	20	48.30	59.40	65.60	57.60	70.50	76.20	52.09	64.33	69.92
	50	49.10	62.10	66.80	59.20	71.80	77.50	53.06	66.15	71.31

Table 2
Performance (%) of retrieval-augmented suggestions as a function of the number of injected examples (Added Ex.) and the prompting strategy (two-step vs. single-step), evaluated across three instruction-tuned LLMs.

slightly degrades when moving to 50 examples.

Crucially, the interaction between retrieval and two-step prompting is systematic. While retrieval alone significantly boosts recall-oriented metrics (Accuracy@10), the two-step strategy consistently improves ranking quality, as reflected in higher MRR values, especially for larger models.

Model scale remains an important factor, but its relative contribution changes in the presence of retrieval. With sufficient external evidence, smaller models close a large portion of the gap with larger ones, suggesting that retrieval-based few-shot supervision partially compensates for limited parametric capacity.

Overall, these results support the view that retrieval and structured reasoning play complementary roles: retrieval improves coverage and recall by constraining the hypothesis space, while two-step prompting acts as an implicit re-ranking and verification mechanism, improving stability and ranking quality.

Final configuration and official results. Table 2 shows that increasing the number of retrieval-augmented suggestions from $k = 10$ to $k = 50$ yields only modest performance gains, while substantially increasing prompt length and, consequently, inference cost and context pressure. For this reason, in the final submission we set $k = 10$, keeping the retrieval context compact while retaining most of the benefits of few-shot evidence injection. The submitted system therefore combines (i) retrieval-augmented few-shot evidence with $k = 10$, (ii) a single-call two-step prompting strategy, and (iii) explicit probabilistic ranking.

Table 3 reports the official results for CRUCIVERB-IT Task 1. UniTor achieves an Accuracy@1 of 0.69, an Accuracy@10 of 0.83, and an MRR of 0.72, substantially outperforming both the shared-task baseline and the average score across submitted runs.

The baseline provided by the task organizers frames clue answering as a pure information retrieval problem: for each test clue, it retrieves the top-10 most similar training clues using BM25 and returns the corresponding answers as candidates. While this approach performs well for near-duplicate clues, it struggles when the clue–solution relationship relies on paraphrasing, lexical substitution, polysemy, or crossword-specific conventions.

UniTor improves substantially over this baseline by combining retrieval with explicit LLM-based selection. Retrieved items act as length-compatible evidence and lightweight supervision, while the two-step prompting strategy enforces a clear separation between candidate exploration and constraint-aware re-ranking. This combination yields a marked improvement both in top-1 accuracy ($+0.29$

Table 3
Top-performing systems in the official CRUCIVERB-IT Task 1 evaluation.

System	Accuracy@1	Accuracy@10	MRR
UniTor	**0.69**	**0.83**	**0.72**
Second-best system	0.58	0.75	0.63
Average score (all runs)	0.52	0.70	0.57
Baseline	0.40	0.62	0.46

absolute over the baseline) and in ranking quality (MRR $+0.26$), indicating that the gains stem not only from improved recall but also from more reliable ordering of plausible candidates.

Beyond outperforming the baseline, UniTor shows a clear margin over the second-best submitted system across all evaluation metrics (Table 3). In particular, UniTor improves Accuracy@1 by $+0.11$ absolute and MRR by $+0.09$, demonstrating that the advantage is not limited to higher recall at larger cutoffs, but reflects consistently better placement of the correct solution among the top-ranked candidates.

This performance gap is noteworthy given that all systems operate under the same constraints, namely the absence of grid information and the availability of only the target answer length. The results suggest that UniTor's advantage does not arise solely from stronger retrieval, but from the interaction between retrieval and LLM-based selection: the two-step prompting strategy explicitly separates candidate generation from constraint-aware re-ranking, leading to more stable and discriminative rankings.

Compared to the average score across all submitted runs, UniTor achieves gains of $+0.17$ in Accuracy@1 and $+0.15$ in MRR. This consistent margin indicates that the proposed architecture generalizes beyond a single favorable configuration and that its components contribute synergistically rather than in isolation. Our ablation studies further suggest that the strongest results are obtained when the two-step probabilistic prompting strategy is paired with a highly capable LLM (in our case, GLM). While GPT-OSS-based configurations already outperform the baseline and most competing systems, the stronger reasoning and selection capabilities of GLM appear to amplify the benefits of retrieval-augmented evidence and explicit re-ranking. We hypothesize that a GPT-OSS-based variant using the same final configuration ($k = 10$, two-step reasoning, probabilistic output) would still yield competitive results, albeit with a smaller margin over the second-best system, consistent with the trends observed in our validation experiments.

5. Conclusions

In this paper we presented UniTor, an LLM-based system for Italian crossword clue answering submitted to the CRUCIVERB-IT Task 1 at EVALITA 2026. UniTor approaches clue solving as a constrained ranking problem and combines three complementary ingredients: retrieval-augmented few-shot evidence, a structured two-step reasoning strategy implemented within a single LLM call, and explicit probabilistic ranking of candidate solutions. Importantly, the system operates without any task-specific fine-tuning, relying exclusively on prompt design, retrieval grounding, and model-internal reasoning.

Our experimental analysis demonstrates that each component plays a distinct and synergistic role. Retrieval-augmented suggestions provide a strong inductive bias toward crossword-consistent lexical choices and yield large gains in recall-oriented metrics even with a small number of retrieved examples. Two-step prompting improves ranking stability and discrimination among highly plausible candidates, particularly for larger models, by explicitly separating candidate exploration from constraint-aware selection. Explicit probability assignment further enhances ranking quality, especially in combination with structured reasoning, supporting the view that distributional outputs mitigate mode collapse and improve relative ordering in ambiguous settings.

On the official Cruciverb-IT Task 1 evaluation, UniTor achieves state-of-the-art performance, substantially outperforming both the BM25-based baseline and all competing systems across Accuracy@1, Accuracy@10, and MRR. The large margin over the baseline confirms that pure retrieval is insufficient to solve crossword clues reliably, while the consistent gap over the second-best system indicates that the gains are not merely due to improved recall, but to more effective ranking and selection. Ablation results further show that retrieval-based grounding allows smaller models to close a significant portion of the performance gap with larger ones, while the strongest results are obtained when retrieval, two-step reasoning, and probabilistic ranking are paired with a highly capable LLM.

This work focuses exclusively on clue answering in isolation and does not address grid-level crossword solving. Integrating UniTor into a full crossword solver that exploits crossing constraints and global grid consistency represents a natural next step, potentially allowing the ranking scores produced by UniTor to act as soft constraints in a global optimization framework. More broadly, our results suggest that for knowledge-intensive, weakly specified NLP tasks, carefully designed prompting and retrieval strategies can rival or surpass more complex training-based approaches. We hope that UniTor provides a useful reference point for future work on crossword solving and on structured, retrieval-grounded reasoning with large language models.

Acknowledgments

The authors acknowledge financial support from the PNRR MUR project PE0000013-FAIR. Moreover, the authors acknowledge support from Project ECS 0000024 Rome Technopole, - CUP B83C22002820006, NRP Mission 4 Component 2 Investment 1.5, Funded by the European Union - NextGenerationEU.

Declaration on Generative AI

Parts of the writing and editing process for this manuscript were supported by the use of generative AI tools, specifically OpenAI's ChatGPT. These tools were employed to enhance clarity and correct spelling. All AI-assisted content was carefully reviewed and validated by the authors to ensure accuracy, originality, and compliance with ethical and scientific standards. The authors bear full responsibility for the final content.

References

[1] C. Ciaccio, G. Sarti, A. Miaschi, F. Dell'Orletta, M. Nissim, Cruciverb-it @ evalita 2026: Overview of the crossword solving in italian task, in: Proceedings of the Ninth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2026), CEUR.org, Bari, Italy, 2026.

[2] F. Cutugno, A. Miaschi, A. P. Aprosio, G. Rambelli, L. Siciliani, M. A. Stranisci, Evalita 2026: Overview of the 9th evaluation campaign of natural language processing and speech tools for italian, in: Proceedings of the Ninth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2026), CEUR.org, Bari, Italy, 2026.

[3] C. Ciaccio, G. Sarti, A. Miaschi, F. Dell'Orletta, Crossword space: Latent manifold learning for Italian crosswords and beyond, in: C. Bosco, E. Jezek, M. Polignano, M. Sanguinetti (Eds.), Proceedings of the Eleventh Italian Conference on Computational Linguistics (CLiC-it 2025), CEUR Workshop Proceedings, Cagliari, Italy, 2025, p. 245–255. URL: https://aclanthology.org/2025.clicit-1.26/.

[4] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling laws for neural language models, 2020. URL: https://arxiv.org/abs/2001.08361. arXiv:2001.08361.

[5] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc,

A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, L. Sifre, Training compute-optimal large language models, 2022. URL: https://arxiv.org/abs/2203.15556. arXiv:2203.15556.

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL: https://arxiv.org/abs/2005.11401. arXiv:2005.11401.

[7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, 2023. URL: https://arxiv.org/abs/2201.11903. arXiv:2201.11903.

[8] 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, 2023. URL: https://arxiv.org/abs/2203.11171. arXiv:2203.11171.

[9] J. Zhang, S. Yu, D. Chong, A. Sicilia, M. R. Tomz, C. D. Manning, W. Shi, Verbalized sampling: How to mitigate mode collapse and unlock llm diversity, 2025. URL: https://arxiv.org/abs/2510.01171. arXiv:2510.01171.

[10] M. L. Littman, G. A. Keim, N. Shazeer, A probabilistic approach to solving crossword puzzles, Artificial Intelligence 134 (2002) 23–55. URL: https://www.sciencedirect.com/science/article/pii/S000437020100114X. doi:https://doi.org/10.1016/S0004-3702(01)00114-X.

[11] G. Angelini, M. Ernandes, M. Gori, Webcrow: A web-based crosswords solver, in: M. Maybury, O. Stock, W. Wahlster (Eds.), Intelligent Technologies for Interactive Entertainment, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 295–298.

[12] P. Basile, M. de Gemmis, P. Lops, G. Semeraro, Solving a complex language game by using knowledge-based word associations discovery, IEEE Transactions on Computational Intelligence and AI in Games 8 (2014) 1–1. doi:10.1109/TCIAIG.2014.2355859.

[13] G. Barlacchi, M. Nicosia, A. Moschitti, Learning to rank answer candidates for automatic resolution of crossword puzzles, in: R. Morante, S. W.-t. Yih (Eds.), Proceedings of the Eighteenth Conference on Computational Natural Language Learning, Association for Computational Linguistics, Ann Arbor, Michigan, 2014, pp. 39–48. URL: https://aclanthology.org/W14-1605/. doi:10.3115/v1/W14-1605.

[14] A. Moschitti, M. Nicosia, G. Barlacchi, SACRY: Syntax-based automatic crossword puzzle resolution sYstem, in: H.-H. Chen, K. Markert (Eds.), Proceedings of ACL-IJCNLP 2015 System Demonstrations, Association for Computational Linguistics and The Asian Federation of Natural Language Processing, Beijing, China, 2015, pp. 79–84. URL: https://aclanthology.org/P15-4014/. doi:10.3115/v1/P15-4014.

[15] S. Saha, S. Chakraborty, S. Saha, U. Garain, Language models are crossword solvers, in: L. Chiruzzo, A. Ritter, L. Wang (Eds.), Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), Association for Computational Linguistics, Albuquerque, New Mexico, 2025, pp. 2074–2090. URL: https://aclanthology.org/2025.naacl-long.104/. doi:10.18653/v1/2025.naacl-long.104.

[16] A. Sadallah, D. Kotova, E. Kochmar, What makes cryptic crosswords challenging for LLMs?, in: O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, S. Schockaert (Eds.), Proceedings of the 31st International Conference on Computational Linguistics, Association for Computational Linguistics, Abu Dhabi, UAE, 2025, pp. 5102–5114. URL: https://aclanthology.org/2025.coling-main.342/.

[17] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, Z. Liu, M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2025. URL: https://arxiv.org/abs/2402.03216. arXiv:2402.03216.

[18] S. Robertson, H. Zaragoza, The probabilistic relevance framework: Bm25 and beyond, Found. Trends Inf. Retr. 3 (2009) 333–389. URL: https://dl.acm.org/doi/abs/10.1561/1500000019.

[19] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, al., The llama 3 herd of models, 2024. URL: https://arxiv.org/abs/2407.21783. arXiv:2407.21783.

[20] OpenAI, S. Agarwal, L. Ahmad, J. Ai, S. Altman, A. Applebaum, al., gpt-oss-120b & gpt-oss-20b

model card, 2025. URL: https://arxiv.org/abs/2508.10925. arXiv:2508.10925.
[21] GLM-4.5 Team, A. Zeng, X. Lv, Q. Zheng, Z. Hou, B. Chen, C. Xie, C. Wang, D. Yin, H. Zeng, al., Glm-4.5: Agentic, reasoning, and coding (arc) foundation models, 2025. URL: https://arxiv.org/abs/2508.06471. arXiv:2508.06471.

A. Appendix

This appendix reports the prompt templates used to query the Large Language Models in our experiments. The templates are systematically constructed by varying three orthogonal dimensions: (i) the prompt structure, distinguishing between a single-step (plain) formulation and a two-step formulation; (ii) the inclusion or exclusion of explicit scoring, where the model is instructed to assign confidence or probability values to candidate solutions; and (iii) the maximum number of requested suggestions, set to 0, 10, 20, or 50. Together, these dimensions define the set of prompt configurations evaluated in the study. This template corresponds to a single-step prompt setting in which no suggestions are provided and no explicit scoring of candidate answers is requested:

```
Sei un esperto risolutore di cruciverba in italiano.
Riceverai in input una definizione. Il tuo compito e' fornire fino a <N> possibili soluzioni, ognuna
     composta esattamente da <L> caratteri.
Scrivi ogni proposta su una nuova riga, senza numerarle e senza aggiungere commenti.
Definizione: <D>
```

This template corresponds to a single-step prompt setting without retrieval-based suggestions, in which the scoring of candidate answers is explicitly requested.

```
Sei un esperto di cruciverba in italiano.
Riceverai una definizione e dovrai fornire fino a <N> possibili soluzioni, indicando per ciascuna una
     probabilità compresa tra 0 e 1 (numero decimale) che sia quella corretta.

Ogni soluzione deve:
 - essere composta esattamente da <L> caratteri
 - essere riportata su una nuova riga nel formato:
   <soluzione>, <probabilità>

Non aggiungere numerazioni, commenti o spiegazioni. Ordina tutte le proposte in ordine decrescente di
     probabilità.
Definizione: <D>
```

This template represents a single-step prompt with up to R retrieval-based suggestions and explicit candidate scoring.

```
Sei un esperto di cruciverba in italiano.
Riceverai:
 - una definizione principale, per la quale dovrai proporre fino a <N> soluzioni possibili, indicando
      per ciascuna una probabilità compresa tra 0 e 1 (numero decimale) che sia corretta;
 - fino a <R> coppie (parola - definizione), da considerare come informazioni aggiuntive potenzialmente
       utili.

Ogni soluzione deve:
 - essere composta esattamente da <L> caratteri
 - essere riportata su una nuova riga nel formato:
<soluzione>, <probabilità>

Non aggiungere numerazioni, commenti o spiegazioni.
Ordina le proposte in ordine decrescente di probabilità.
Definizione: <D>
```

```
Informazione aggiuntiva: <up to R word/definition pairs>
```

This template corresponds to a two-step prompt configuration with retrieval-based suggestions (with a maximum number of S), in which the explicit scoring of candidate answers is required.

Each prompt enforces a two-phase generation process: (i) an exploratory stage (`candidati_iniziali`), in which up to N (set to 10) semantically plausible candidates are proposed without enforcing length constraints; and (ii) a selection stage (`definitivi`), which applies normalization, exact length filtering, and optional schema constraints, followed by re-ranking of the remaining candidates. Retrieved clue–solution pairs (up to R) are injected as lightweight evidence to support candidate generation and selection.

```
Sei un esperto risolutore di cruciverba in lingua italiana.
Riceverai:
- una definizione testuale;
- un numero di caratteri (lunghezza esatta, solo lettere A-Z);
- fino a <R> definizioni (parola - definizione), che potrebbero essere utili.

## STRATEGIA A DUE FASI (con output doppio)
Genera due liste in sequenza:

1) FASE 1 - CANDIDATI_INIZIALI (brainstorming)
   - Genera fino a <N> parole plausibili semanticamente rispetto alla definizione (sinonimi, iperonimi,
        tecnicismi, locuzioni normalizzate).
   - Possono includere lunghezze errate rispetto a `[[LUNGHEZZA]]`. Non scartarle qui: questa è una
       fase esplorativa.
   - Normalizza comunque tutte le voci (vedi regole).
   - Calcola `probabilita` secondo i pesi indicati (sotto).
   - Ordina per probabilità decrescente e rimuovi duplicati.

2) FASE 2 - DEFINITIVI (filtro & raffinamento)
   - Prendi solo i candidati della FASE 1 che:
     - hanno `LEN(normalizzata) == [[LUNGHEZZA]]`;
   - Ispirati ai candidati della FASE 1 e alla fonti per aggiungere:
     - sinonimi piu' precisi/usuali in ambito cruciverbistico,
     - varianti morfologiche corrette,
     - lemmi preferiti (singolare/maschile-forma base, salvo diversa indicazione).
   - Escludi qualsiasi voce che non rispetti lunghezza e schema.
   - Ricalcola le `probabilita` con gli stessi pesi e ordina.
   - Se nessuna voce è valida, restituisci `"definitivi": []`.

## NORMALIZZAZIONE (sempre, in entrambe le fasi)
- Tutto in MAIUSCOLO.
- Rimuovi accenti, apostrofi, spazi e trattini.
- Ammesse solo lettere A-Z.
- Il campo `lunghezza` è la lunghezza della parola normalizzata.

## SCORING (per entrambe le fasi)
Assegna `probabilita` in (0,1] come somma ponderata:
- 0.40 -> Schema/Lunghezza
  - Fase 1: +0.40 solo se lunghezza corretta e schema rispettato (se presente); altrimenti 0 su questa
      quota.
  - Fase 2: requisito obbligatorio (altrimenti la voce non entra nei `definitivi`).
- 0.35 -> Attinenza semantica alla definizione.
- 0.25 -> Plausibilità cruciverbistica (forma più comune, lemma preferito, uso frequente).

## CHECKLIST (prima di emettere l'output)
- [ ] Tutte MAIUSCOLE e normalizzate
- [ ] `lunghezza` corrisponde alla parola normalizzata
- [ ] In FASE 2: lunghezza corretta e schema rispettato
- [ ] Nessun duplicato
- [ ] Nessun carattere non ammesso
- [ ] `probabilita` in (0,1]

## DIVIETI
- Non aggiungere testo esplicativo fuori dallo JSON richiesto.
- Non includere parole troncate o con caratteri non ammessi.
```

```
- Se la definizione ha “?” -> trattala come definizione figurata/gioco di parole.
- Se è indicato “abbr.” o “sigla” -> proponi solo abbreviazioni/sigle.

## IMPORTANTE - CASI DA 2 LETTERE (Regole d’Oro)

Se ‘[[LUNGHEZZA]] == 2‘, non inventare nulla di semantico o fantasioso.
Quasi sempre si tratta di giochi di lettere o sigle.
Segui rigorosamente queste 10 REGOLE D’ORO:

1. "Inizio / Prime / Principio di ..." -> usa le prime due lettere.
2. "Fine / Ultime / Termine di ..." -> usa le ultime due lettere.
3. "Limiti / Confini / Estreme di ..." -> prendi prima + ultima lettera.
4. "In mezzo a / Nel centro di ..." -> prendi le due centrali (o la coppia interna evidente).
5. "Gemelle / Uguali / Doppie ..." -> prendi la lettera raddoppiata presente (es. “NN”, “LL”).
6. "Consonanti / Vocali di ..." -> conserva solo consonanti o solo vocali del termine.
7. "Sigla / Targa / Provincia / Simbolo" -> scegli abbreviazioni ufficiali o codici (AR, LI, PD, BO, AI
     , UV, KO, OK...).
8. "Iniziali di ..." / "(iniz.)" -> usa le prime lettere dei nomi indicati.
9. "Senza / Fra / Tra ..." -> togli o confronta lettere esplicitamente tra le parole.
10. Definizione secca (tipo “Preposizione”, “Negazione”, “Risposta affermativa”) -> usa lemmi di 2
     lettere noti:
    IN, NO, SI, SU, DA, DI, RE, RA, TO, OR, ME, IO, ecc.

## FORMATO DI OUTPUT
Un unico oggetto JSON con due array in questo ordine:
- ‘candidati_iniziali‘ -> fino a <N> candidati (possono includere lunghezze errate).
- ‘definitivi‘ -> fino a <N> voci filtrate e migliorate (tutte con lunghezza corretta e schema
     rispettato).

Ogni elemento di entrambi gli array deve rispettare questo schema:
{ "parola": "^[A-Z]+$", "lunghezza": <int>, "probabilita": <decimal number from 0.0 to 1.0> }

# Response Formats
## crossword_double
// Oggetto con le due liste: brainstorming (fase 1) e soluzione finale (fase 2).
{
  "type": "object",
  "properties": {
    "candidati_iniziali": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "parola": { "type": "string", "pattern": "^[A-Z]+$" },
          "lunghezza": { "type": "integer", "minimum": 1 },
          "probabilita": { "type": "number", "minimum": 0, "maximum": 1 }
        },
        "required": ["parola", "lunghezza", "probabilita"],
        "additionalProperties": false
      }
    },
    "definitivi": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "parola": { "type": "string", "pattern": "^[A-Z]+$" },
          "lunghezza": { "type": "integer", "minimum": 1 },
          "probabilita": { "type": "number", "minimum": 0, "maximum": 1 }
        },
        "required": ["parola", "lunghezza", "probabilita"],
        "additionalProperties": false
      }
    }
  },
  "required": ["candidati_iniziali", "definitivi"],
  "additionalProperties": false
}
Definizione: <D>
Lunghezza: <L>
Informazione aggiuntiva: <up to R word/definition pairs>
```