FFT-UniBa at Cruciverb-IT: Special Length Tokens and CSP for Italian Crossword Solving

Andrea Porcelli[1,†], Filippo Di Gravina[1,†], Emanuele Fontana[1], Mattia Curri[1,*] and Francesco Damiano Di Gregorio[1]

[1]Department of Computer Science, University of Bari Aldo Moro, Via Edoardo Orabona 4, 70125, Bari, Italy

Abstract

Cruciverb-IT [1], a shared task proposed at EVALITA 2026 [2], aims to encourage research in automated crossword solving systems by providing a challenging testbed for developing and evaluating systems focused on crossword puzzle solving. The proposed sub-tasks are the following: answering clues extracted from Italian crosswords and autonomously solving Italian crossword grids. To succeed in crossword solving, models must acquire meta-linguistic abilities beyond pure semantic understanding, like counting individual characters or enforcing strict length and pattern constraints typical of crossword answers. In this paper, we present our approach to both sub-tasks, which involves fine-tuning an Italian T5 model with special tokens to represent answer lengths and employing a constraint satisfaction problem (CSP) solver for grid filling. The system demonstrates competitive performance on both tasks, showcasing the effectiveness of our methods in mitigating the challenges posed by crossword puzzles. The code, and any additional resources used in our contribution can be found at https://github.com/mattiacurri/fftuniba-cruciverbit.

Keywords

Language Games, Crossword Solving, Constraint Satisfaction Problem, Natural Language Processing, Information Retrieval, Computational Linguistics

1. Introduction

Crosswords Puzzles are the most famous language games played around the world [3]. Among popular language games, crossword puzzles stand out as particularly challenging, demanding not only linguistic competence but also extensive world knowledge, cultural awareness, and lateral thinking skills [4]. Despite the impressive advancements in Large Language Models (LLMs), their performance on language games such as crosswords remains limited, especially in morphologically rich and less-resourced languages like Italian [5, 6, 7], where crosswords are characterized by deeper and more ambiguous references to socio-cultural and political contexts [8]. To succeed in solving crosswords, models must acquire meta-linguistic abilities beyond pure semantic understanding, such as counting individual characters or enforcing strict length and pattern constraints typical of crossword answers [9, 10, 11]. Because common tokenization schemes operate at word, subword, or byte-level granularity, character-level properties are not always preserved or directly accessible, so models cannot rely on token counts to satisfy orthographic constraints [12]. To be robust, systems should either learn to incorporate explicit character-aware representations, reconstruct character sequences from tokens, or be trained with character-level supervision or auxiliary objectives that expose and enforce these constraints [13]. To evaluate these capabilities in a standardized setting, the Cruciverb-IT shared task [1] was proposed within the EVALITA 2026 campaign [2]. The challenge establishes a benchmark for Italian crossword solving, dividing the problem into two distinct sub-tasks: retrieving the correct answer for a given clue (sub-task 1) and filling complete crossword grids (sub-task 2). In this work, we present

EVALITA 2026: 9th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, Feb 26 - 27, Bari, IT

*Corresponding author.

†These authors contributed equally.

a.porcelli34@studenti.uniba.it (A. Porcelli); f.digravina2@studenti.uniba.it (F. D. Gravina); e.fontana7@studenti.uniba.it (E. Fontana); m.curri8@studenti.uniba.it (M. Curri); f.digregorio16@studenti.uniba.it (F. D. D. Gregorio)

mattiacurri.github.io (M. Curri)

0009-0008-7400-5820 (E. Fontana); 0009-0001-0546-7676 (M. Curri)

our contribution to this challenge, choosing to explore the strategy of explicit length conditioning. The paper is structured as follows: Section 2 covers related work, followed by a description of the challenge sub-tasks in Section 3. The proposed approach and experimental framework are detailed in Sections 4 and 5. Main findings are discussed in Section 6, with an ablation analysis presented in Section 7. Conclusion, limitation and future work are reported in Section 8.

2. Related Work

The field of automatic crossword puzzle resolution was pioneered by Proverb [14], the first system designed to reach human-like performance using domain-specific expert modules and a large database of clue-answer pairs. Following Proverb, WebCrow [8] was developed as the first solver for non-English crosswords, specifically targeting the Italian language by utilizing the Web as its primary knowledge base. WebCrow represents the solving process as a Probabilistic Constraint Satisfaction Problem (PCSP), with a new web module which searches the web for snippets that may contain answers [15]. Another landmark system, Dr.Fill [16], treats crossword filling as a Weighted-CSP, where constraint violations are weighted and tolerated. Barlacchi et al. [17] apply a BM25 retrieval model to generate clue lists similar to the query clue from historical clue-answer database, where the generated clues get further refined through application of re-ranking models. Severyn et al. [18] introduce a distributional neural network to compute similarities between clues trained over a large scale dataset of clues that they introduce. Barlacchi et al. [3] presented SACRY, a syntax-based automatic crossword resolution system that extends the WebCrow architecture by integrating structural kernels and learning-to-rank models for effective clue reranking and answer aggregation. Rozner et al. [10] demonstrates that while a standard T5 baseline struggles at solving crosswords, a curriculum learning strategy involving synthetic wordplay tasks significantly improves performance, though the model remains unable to fully generalize compared to human experts. Wallace et al. [9] introduced the Berkeley Crossword Solver, a state-of-the-art system that integrates neural bi-encoder QA models with loopy belief propagation and local search. This approach achieved 82% perfect puzzle accuracy on New York Times crosswords, significantly outperforming Dr.Fill and top human competitors. Ciaccio et al. [4] recently proposed a contrastive learning framework for Italian crosswords, using an Asymmetric Dual Encoder architecture to model a shared latent space between clues and their solutions, significantly outperforming standard retrieval baselines in the Italian language.

3. Task Description

Sub-task 1 Given a set of clues $C = \{c_i\}_{i=1}^{n}$ and the expected answer length $\ell_i = |s_i|$ for each clue, the goal is to produce, for every c_i, a ranked list of candidate solutions $\hat{S}_i = (\hat{s}_{i,1}, \ldots, \hat{s}_{i,m})$ such that each candidate satisfies the length constraint $|\hat{s}_{i,j}| = \ell_i$.

Sub-task 2 Given a set of crossword instances $\mathcal{G} = \{G_t\}_{t=1}^{k}$, each instance G_t consists of a rectangular layout (a matrix of cells) where each cell is either blocked (black) or fillable, and a set of slots V_t described by a clue and by its placement metadata: starting coordinates (x_v, y_v) and direction $d_v \in \{\textsc{across}, \textsc{down}\}$. The objective is to assign a word w_v to every slot $v \in V_t$ such that $|w_v|$ matches the slot length induced by the layout, and all crossing constraints are satisfied, i.e., letters in intersecting slots agree on their shared cells.

4. Proposed Approach

Sub-task 1

The adopted model is *it5-efficient-small-el32*[1] [19], an Italian-trained variant of T5 [20] based on the Efficient T5 architecture [21]. Since matching the exact length of the ground truth is a primary challenge in crossword solving, we introduce a *Dual-Constraint* mechanism to explicitly bounds the generation process. Let $\mathcal{C}$ denote the clue and S the target solution of length L. We construct the input sequence X by embedding the target length alongside the clue:

$$X = \texttt{indovinello:} \oplus \mathcal{C} \oplus \texttt{[SL=L]} \tag{1}$$

Here, $\oplus$ represents string concatenation. The token '`[SL=L]`' (Start Length) serves as an explicit prompt, instructing the model on the required character count before generation begins. Similarly, we modify the target output sequence Y to include a termination token '`[EL=L]`' (End Length):

$$Y = \texttt{[SL=L]} \oplus S \oplus \texttt{[EL=L]} \tag{2}$$

For every length encountered in the training set, two tokens (one for each type) are added directly to the model's tokenizer, and their embeddings are initialized using a normal distribution. This technique acts effectively as a structural "zipper": the generation is anchored between a specific start token and a specific end token, forcing the model to plan the word construction within fixed boundaries. As an example, consider the clue "Capital of Italy" with a required solution length of 4; the input provided to the model is `indovinello: Capital of Italy [SL=4]`, and the target output on which it is trained is `[SL=4] Rome [EL=4]`. This approach differs fundamentally from simply filtering outputs during inference (e.g., masking logits to forbid the EOS token until length L is reached). If a model is not trained to anticipate the end of a word, forcing it to continue or stop artificially often leads to coherent but incorrect completions. By training with these tokens, the model learns to internalize the length constraint: the probability distribution of the generated tokens naturally converges towards the end-token '`[EL=L]`' exactly when the word is complete. Evaluating whether the model actually benefits from learning the placement of the final token is left as a future ablation study.

Sub-task 2

We formulate the sub-task as a Constraint Satisfaction Problem (CSP), formally defined by the tuple $\langle V, D, \mathcal{C} \rangle$.

- V (Variables): The set of all empty slots (definitions) in the grid.
- D (Domains): For each slot $v \in V$, the domain D_v is the ranked list of candidate words composed of the top-K predictions generated by our fine-tuned model (Sub-task 1), ranked by perplexity (lower is better), and a set of K' auxiliary words retrieved from a standard dictionary[2]. These are used only if valid model candidates are exhausted and are strictly filtered to match existing letters in the grid.
- $\mathcal{C}$ (Constraints): The set of intersection rules ensuring that horizontal and vertical words share the same letter at crossing points.

Crucially, we set $K' \ll K$ to ensure that the system behaves as a neural solver, relying on semantic understanding rather than brute-force dictionary lookups. The solving algorithm proceeds in two stages:

1. **Initialization (Seeding):** Sorting answers by perplexity we start from different positions i.e. we sets the two best words predicted by the model for the clues 1,3 as anchors. We generate multiple seed configurations, where each seed fixes a high-confidence model prediction into the grid to serve as an anchor.

[1] https://huggingface.co/gsarti/it5-efficient-small-el32
[2] https://github.com/napolux/paroleitaliane/blob/main/paroleitaliane/parole_uniche.txt

2. **Recursive Backtracking:** From each seed, we perform a Depth-First Search (DFS). At each node, the algorithm attempts to fill the next slot by choosing the most probable compatible word from D_v. If a conflict arises later, it backtracks to try the next candidate.

To balance solution quality with computational efficiency, we employ an adaptive resource allocation strategy. Larger grids induce a larger search space; therefore, we scale the number of candidates (K, K') and the number of initial seeds based on the grid dimension n. The number of candidates drawn from dictionary are limited, to avoid the degeneration to a pure dictionary-based CSP solver, which would not follow the challenge constraints. The specific schedule is detailed in Table 1.

Table 1
Adaptive hyperparameters based on grid size n. K denotes model candidates, K' dictionary candidates.

Grid Size (n)	Initial Seeds	Model (K)	Dict (K')
Small ($n \leq 6$)	100	100	3
Medium ($7 \leq n \leq 8$)	500	500	5
Large ($9 \leq n \leq 10$)	1000	600	10
Extra Large ($n > 10$)	2000	600	10

Here we discuss the complexity of the algorithm. Let $m = |V|$ be the number of clues (slots) to be filled in the crossword. For each clues, the algorithm disposes of two sets of candidates: at most K words originating from the model output and at most K' words from the dictionary. The resolution proceeds via a Depth-First Search (DFS) with backtracking, which iteratively assigns an answer to a clue and incrementally verifies compatibility constraints.

Time Complexity. The computational cost associated with a single node in the search tree, which represents a specific partial assignment of a candidate word to a target slot, is dominated by the consistency verification. This operation checks the intersection constraints against the previously assigned words in the grid. Since verifying a candidate requires checking character conflicts with at most m intersecting slots (where m is the total number of clues), this step requires polynomial time and is upper-bounded by $O(m)$. In the worst-case scenario, assuming no pruning occurs, the algorithm explores up to $K + K'$ alternatives for each of the m slots. This results in a search tree with a maximum branching factor of $K + K'$ and a depth equal to m. Consequently, the total number of explorable nodes is of the order $O((K + K')^m)$. Combining the node count with the verification cost, the worst-case time complexity is $O(m \cdot (K + K')^m)$. In dominant asymptotic notation, this simplifies to $O((K + K')^m)$. In practice, to manage this complexity, we enforce the budgets described in Table 1. Table 2 reports the average runtime observed in our experiments per grid size. It is important to note that these timings refer strictly to the solving phase; they exclude the candidate generation preprocessing (which required approximately 10 minutes to generate 600 candidates using a beam search of width 1000).

Table 2
Average runtime (in seconds) for grid solving by size, using 600 candidates. Experiments were conducted on a Ryzen 7 7700 CPU. Timings exclude the candidate generation preprocessing phase.

Grid Size	Min (s)	Max (s)	Mean (s)
5×5	0.21	4.18	1.75
7×7	4.01	26.70	13.01
9×9	14.17	55.29	30.05
11×11	63.92	155.83	102.03
13×13	151.09	754.89	309.99

Space Complexity. The space complexity is $O(m)$, determined primarily by the recursion stack used by the backtracking algorithm and the data structure storing the current partial assignment (i.e.,

the mapping of words to slots). Since the algorithm operates depth-first, the maximum depth of the recursion stack corresponds exactly to the total number of clues m to be filled.

5. Experimental Setup

In this section, we detail the dataset, evaluation metrics, baselines, training configuration, and the specific settings used to assess our methodology.

Dataset Description and Data Augmentation

The dataset proposed by the challenge organizers relies on the ItaCW crossword dataset [22], significantly enriched with a collection of additional clue-solution pairs retrieved from the web. As summarized in Table 3, the data is organized to support both sub-tasks. For the first sub-task, the overall dataset comprises approximately 410,000 clue-answer pairs. These data are partitioned according to a standard scheme: 90% for the training set, 5% for validation, and 5% for testing. Regarding the second sub-task, crossword grids were generated by employing a constraint-driven, search-based construction algorithm. This algorithm was designed to populate predefined crossword layouts using exclusively valid words from the list of answers contained in the aforementioned data splits. The generation process yielded crosswords of varying sizes, resulting in 500 grids for the training set, 50 for the validation set, and 50 for the test set. To improve the vocabulary coverage, we integrate a dictionary dataset[3] $\mathcal{D} = \{(d_i, w_i)\}$, containing approximately 313,000 definition-word pairs. These pairs undergo the same preprocessing (Equation 1 and 2) as the crossword clues. This augmentation could allow the model to learn the association between semantic descriptions and structural constraints on a much wider scale, generalizing its ability to define words of specific lengths beyond the stylistic syntax typical of crossword puzzles.

Table 3
Overview of the Challenge Dataset and Sub-tasks

Feature	Description/Value
Sub-task 1: Clue-Solution Pairs	
Total Size	$\approx$ 410,000 pairs
Data Splits	Training (90%), Validation (5%), Test (5%)
Clue Categories	Wordplays, cryptic clues, named entities initials, fill-the-fill, etc.
Sub-task 2: Crossword Grids	
Generation Method	Constraint-driven, search-based construction algorithm
Training Set	500 grids
Validation Set	50 grids
Test Set	50 grids

Evaluation

To assess performance, we employ the official challenge metrics for each sub-task:

Sub-task 1

- **Accuracy@k** (with $k = 1, 10$): The accuracy in retrieving the correct solution word given the clue, considering the top-k system candidates.
- **Mean Reciprocal Rank (MRR)**: The average of the reciprocal ranks of the first relevant item across all clues.

[3] https://huggingface.co/datasets/mik3ml/italian-dictionary

To isolate the impact of special tokens described in Section 4, we use two modes:

1. *Constrained*: Output is filtered to include only answers matching the specified length.
2. *Unconstrained*: All generated answers are considered regardless of length.

Sub-task 2

- **Character Accuracy**: The percentage of correct characters inserted in the correct slots.
- **Word Accuracy**: The percentage of correct words inserted in the correct slots.
- **Full Grid Accuracy**: The percentage of completely solved grids.

We evaluate the algorithm by varying the number of generated candidates and testing models trained for different epoch counts. For both sub-tasks, we assess the impact of external knowledge by enabling or disabling dictionary augmentation to determine if it improves final performance.

Baselines

We compare our approach against the standard baselines provided for the tasks.

Sub-task 1 The problem is treated as an Information Retrieval (IR) task. Given a test clue $c_i \in C_{test}$, the system computes the BM25 similarity score against all training clues C_{train}. The top ten most similar clues are selected, and their corresponding answers are extracted as candidates.

Sub-task 2 This baseline combines the aforementioned ranker with a module that treats crossword puzzles as a weighted Max-SMT problem [15]. It optimizes for a solution by defining hard constraints (the grid structure) and soft constraints (candidate ranking preferences). Specifically, each clue corresponds to a disjunctive group of grid variables constrained to match candidate answers, combined conjunctively across the grid. The formulation uses the Z3 optimizer[4] [23] with 10 candidates per clue.

Training Configuration

We trained our model using the hyperparameters outlined in Table 4. Due to time and resource constraints, we did not perform extensive hyperparameter optimization.

Table 4
Training hyperparameters and configuration details.

Hyperparameter	Value
Max Input Length	128
Max Target Length	32
Batch Size	32
Gradient Accumulation	None
Warmup Steps	100
Epochs	22–23
Learning Rate	2e-4

6. Results and Discussion

We present a comprehensive analysis of the experimental results obtained for both subtasks.

[4] https://github.com/pncnmnp/Crossword-Solver

Table 5
Sub-task 1 results ordered by MRR. **Bold** indicates best result per metric, underlined indicates second best.

Model	Epochs	Dictionary dataset	Length tokens	Constrained?	Acc@1	Acc@10	MRR
†		-	-	-	**0.69**	**0.83**	**0.72**
it5-efficient-small-el32*	23	✓	✓	✓	0.58	0.75	0.63
†		-	-	-	0.59	0.71	0.62
it5-efficient-small-el32*	22	✓	✓	✓	0.57	0.75	0.62
it5-efficient-small-el32*	23	✓	✓	✗	0.55	0.72	0.60
it5-efficient-small-el32*	22	✓	✓	✗	0.54	0.73	0.60
†		-	-	-	0.51	0.73	0.57
†		-	-	-	0.47	0.67	0.53
†		-	-	-	0.46	0.69	0.52
†		-	-	-	0.43	0.59	0.47
Baseline		-	-	-	0.40	0.62	0.46
†		-	-	-	0.36	0.54	0.41

*Submitted score for the challenge
†Other participants' submission

Sub-task 1 Table 5 reports the performance for Sub-task 1, ranked by Mean Reciprocal Rank (MRR). The proposed approach significantly outperforms the challenge baseline across all metrics. Our best-performing configuration finetuned for 23 epochs with dictionary augmentation and special length tokens achieved an Accuracy@1 of 0.58 and an MRR of 0.63. This represents a substantial margin over the baseline (MRR 0.46) and confirms the efficacy of the it5 architecture for this specific domain with the addition of the special tokens. A crucial finding from Sub-task 1 is the interaction between training-time constraints and inference-time filtering. While strictly enforcing answer length during inference yields a consistent performance boost (approximately +0.03 MRR across configurations), the performance gap between models trained with special tokens and those without them is far larger. This empirical evidence suggests that the explicit tokens ($[SL = L]$ and $[EL = L]$) allow the model to internalize the length constraint, resolving semantic ambiguities and distinguishing between synonyms of different lengths during the generation process itself.

Sub-task 2 Table 6 details the solving performance on complete crossword grids. Our CSP approach demonstrates a decisive advantage over the baseline, achieving a Full Grid Accuracy of 34% compared to the baseline's 8%. An interesting divergence emerges when comparing Sub-task 1 and Sub-task 2 metrics. While the model trained for 23 epochs exhibited superior performance on isolated clues (Sub-task 1), the model trained for 22 epochs proved more effective for full grid solving (34% vs. 28% full grid accuracy). This counter-intuitive result implies that the model trained for 23 epochs may have begun to overfit. Furthermore, integrating the dictionary as a secondary candidate source consistently improved the final solution rate. Although the model successfully handles the vast majority of semantic clues (reflected in the high Character Accuracy of $\sim$92%), the dictionary acts as a critical fallback leading to 1 more full grid solved.

7. Ablation Study

We choose to compare both Sub-task 1 and Sub-task 2 with the best performing run submitted to the challenge.

Sub-task 1 Table 7 presents the breakdown of Sub-task 1 performance. The removal of special length tokens results in a dramatic drop in performance, with MRR falling from 0.63 to 0.43 in the absence of training-time constraints. This confirms that the standard architecture struggles to adhere to

Table 6
Sub-task 2 results ordered by Full Match Accuracy. **Bold** indicates best result per metric, underlined indicates second best.

Algorithm	Epoch	Num Candidates	Dictionary?	Char Accuracy	Word Accuracy	Full Grid Accuracy
Ours*	22	600	✓	0.92	**0.85**	**0.34**
Ours*	22	600	✗	0.92	**0.85**	0.32
Ours*	23	600	✓	**0.93**	0.84	0.28
Ours*	23	600	✗	0.91	0.82	0.22
†				0.82	0.66	0.16
†				0.82	0.67	0.16
Baseline				0.73	0.58	0.08

*Submitted score for the challenge
†Other participants' submission

Table 7
Sub-task 1 ablation study results

Model	Dictionary dataset	Length tokens	Constrained?	Acc@1	Δ Acc@1	Acc@10	Δ Acc@10	MRR	Δ MRR
Best Performer	✓	✓	✓	0.58		0.75		0.63	
	✓	✓	✗	0.55	↓ 0.03	0.72	↓ 0.03	0.60	↓ 0.03
	✗	✓	✓	0.57	↓ 0.01	0.75	- 0.00	0.62	↓ 0.01
	✗	✓	✗	0.54	↓ 0.04	0.72	↓ 0.03	0.59	↓ 0.04
	✓	✗	✓	0.48	↓ 0.10	0.64	↓ 0.11	0.52	↓ 0.11
	✓	✗	✗	0.35	↓ 0.23	0.54	↓ 0.21	0.41	↓ 0.22
	✗	✗	✓	0.38	↓ 0.20	0.54	↓ 0.21	0.43	↓ 0.20
	✗	✗	✗	0.26	↓ 0.32	0.43	↓ 0.32	0.32	↓ 0.31

Table 8
Class Distribution Analysis of the Dataset after labeling

Category	**Count**	**Percentage (%)**
Single Class Counts		
Definition	18,665	89.65
Wordplay	1,961	9.42
Filling the fill	611	2.93
Cryptic	589	2.83
Class Combinations		
Definition	17,822	85.60
Wordplay	1,564	7.51
Filling the fill	389	1.87
Cryptic + Definition	372	1.79
Definition + Wordplay	238	1.14
Definition + Filling the fill	205	0.98
Cryptic + Wordplay	117	0.56
Cryptic	69	0.33
Cryptic + Definition + Wordplay	28	0.13
Filling the fill + Wordplay	14	0.07
Cryptic + Filling the fill	3	0.01

strict character-count constraints without conditioning the model at training time. Interestingly, while applying inference-time constraints to a model trained *without* tokens provides a significant recovery, this "post-hoc" correction remains inferior to the "intrinsic" learning provided by the tokens (MRR 0.60 without filtering). The combination of both mechanisms yields the state-of-the-art result, indicating that they are complementary: the tokens guide the semantic search, while the filter acts as a final safeguard

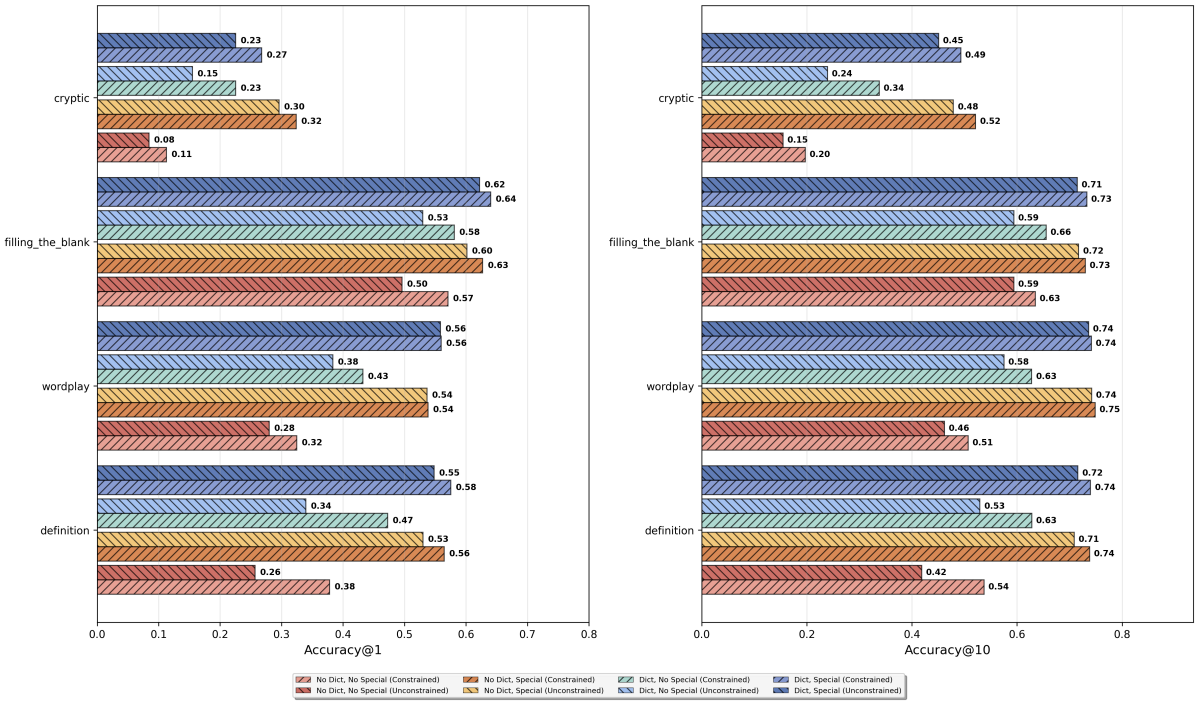

Figure 1: Sub-task 1 classification results

against generation errors. The addition of the dictionary dataset yields only marginal improvements for Sub-task 1 (e.g., +0.01 MRR). To investigate this counterintuitive phenomenon, we labeled the test set clues using Gemini 3 Flash (configured with a temperature of 0.0 and otherwise default parameters) into four distinct labels: `definition`, `wordplay`, `fill-in-the-blank`, and `cryptic`. Table 8 presents the distribution of these labels across the dataset, while Figure 1 illustrates the models' performance across the different clue types. We observe that incorporating the dictionary and special tokens during training yields a negligible improvement in the definition category. Moreover, the gains observed in the fill-in-the-blank and wordplay classes are outweighed by a degradation in performance within the cryptic category. This analysis demonstrates that the original dataset is sufficient for capturing definitions, whereas the inclusion of a dictionary proves detrimental to the model's ability to resolve cryptic clues. Consequently, the dictionary appears to introduce an unnecessary bias toward definitional clues. Therefore, the primary bottleneck in performance stems from complex linguistic reasoning rather than a lack of lexical coverage. However, while this analysis provides valuable insights, these conclusions must be interpreted in light of the inherent limitations and potential biases introduced by the LLM-based generation of the labels. Notably, the LLM occasionally misclassifies fill-in-the-blank clues, a limitation that reinforces some of the findings presented in our work. We release the labeled dataset in our repository, along with the adopted prompt and code to generate it.

Sub-task 2 We analyzed the behavior of the solver by varying the number of candidates k retrieved per clue, ranging from 1 to 600, as illustrated in Figure 2. The candidates are generated from the model trained for 22 epochs, using the same number of beams (1000), that is, our best performer. We observe that beyond $k = 200$, the curve begins to plateau (except for the full match accuracy), indicating that if the correct answer is not within the top 200 predictions, it is unlikely to be retrieved by simply expanding the beam further. This can be due to the fact that the finetuned model is already able to provide good candidates in the first 100-200 candidates. The ablation also highlights the "long tail" effect of the dictionary augmentation: the presence of dictionary candidates consistently maintains a performance gap over the no-dictionary setting. This proves that, while the model handles most of the grid construction, the dictionary candidates are essential for completing the final most difficult

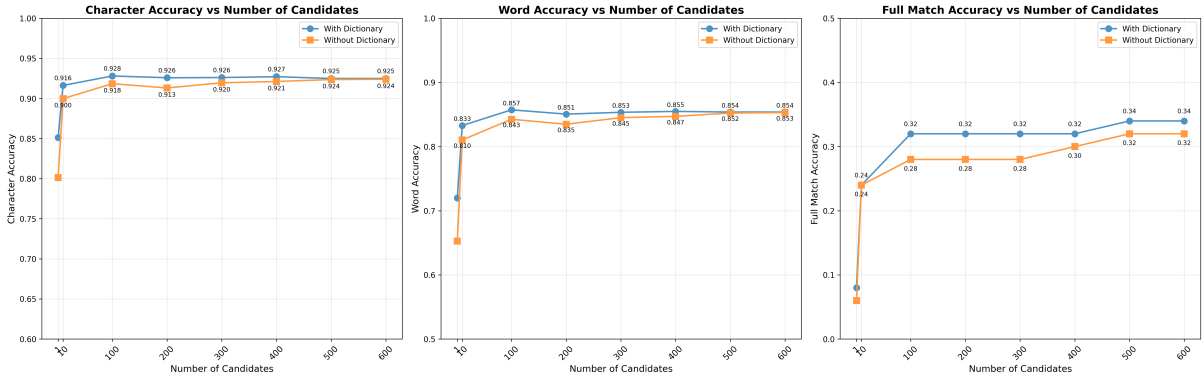

Figure 2: Sub-task 2 ablation study results

intersections, providing the necessary coverage to close the loop on complex grids.

8. Conclusion

In this work, we presented our approach to the Cruciverb-IT shared task at EVALITA 2026, addressing both crossword clue answering and complete grid solving for Italian crosswords. Our methodology combines a fine-tuned sequence-to-sequence model with architectural innovations and a constraint satisfaction problem solver. For Sub-task 1, we introduced a dual-constraint mechanism based on special length tokens '`[SL=L]`' and '`[EL=L]`' that explicitly encode answer length requirements directly into the model's input and output sequences. Our ablation study demonstrates that this token-based approach yields substantial performance gains. The model achieved up to 0.58 Acc@1, 0.75 Acc@10 and 0.63 MRR, substantially outperforming the baseline. For Sub-task 2, we formalized crossword solving as a CSP with adaptive hyperparameters. Our backtracking solver prioritizes model-generated candidates over dictionary fallbacks and employs strict computational budgets to balance solution quality with practical runtime constraints. The solver achieved up to 0.93 character accuracy, 0.85 word accuracy, and 0.34 full grid accuracy on the test set, substantially outperforming the baseline.

Limitations and Future Work While our approach demonstrates strong performance, several directions warrant further investigation. Byte-level tokenization based models such as ByT5 [24], Charformer [25], or MambaByte [26] could provide a viable alternative to our token-based approach by natively operating at character granularity, though this would require pre-training or adapting such architectures for Italian. Remaining within the immediate scope of our proposed system, scaling our methodology to large language models (LLMs) could reveal whether the benefits of explicit structural tokens persist at greater model capacities or whether emergent abilities render such mechanisms redundant, as well as a deeper investigation into the precise impact and dynamics of the added special tokens could yield valuable insights. Beyond architectural considerations, more fine-grained hyperparameter tuning for Sub-task 2 could yield additional performance gains. Furthermore, given that the dataset for Sub-task 2 was generated using clues from Sub-task 1, future evaluations should extend to crosswords created via alternative generation pipelines. Finally, incorporating the crossword grid structure directly into the training objective for Sub-task 1 (e.g., through multi-task learning or synthetic grid completion tasks) could better align the model's predictions with the grid's requirements.

Declaration on Generative AI

During the preparation of this work, the author(s) used Gemini 3 Pro in order to: Paraphrase and reword, Improve writing style, and Grammar and spelling check. After using these tool(s)/service(s), the

author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's 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] G. Barlacchi, M. Nicosia, A. Moschitti, Sacry: Syntax-based automatic crossword puzzle resolution system, in: Proceedings of 53nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Beijing, China, July. Association for Computational Linguistics, 2015.

[4] C. Ciaccio, G. Sarti, A. Miaschi, F. Dell'Orletta, Crossword space: Latent manifold learning for italian crosswords and beyond, in: Proceedings of the Eleventh Italian Conference on Computational Linguistics (CLiC-it 2025), 2025, pp. 245–255.

[5] G. Sarti, T. Caselli, A. Bisazza, M. Nissim, EurekaRebus - verbalized rebus solving with LLMs: A CALAMITA challenge, in: F. Dell'Orletta, A. Lenci, S. Montemagni, R. Sprugnoli (Eds.), Proceedings of the Tenth Italian Conference on Computational Linguistics (CLiC-it 2024), CEUR Workshop Proceedings, Pisa, Italy, 2024, pp. 1202–1208. URL: https://aclanthology.org/2024.clicit-1.132/.

[6] G. Sarti, T. Caselli, M. Nissim, A. Bisazza, Non verbis, sed rebus: Large language models are weak solvers of Italian rebuses, in: F. Dell'Orletta, A. Lenci, S. Montemagni, R. Sprugnoli (Eds.), Proceedings of the Tenth Italian Conference on Computational Linguistics (CLiC-it 2024), CEUR Workshop Proceedings, Pisa, Italy, 2024, pp. 888–897. URL: https://aclanthology.org/2024.clicit-1.96/.

[7] E. Mensa, L. Zane, C. J. Scozzaro, M. Delsanto, T. Milani, D. P. Radicioni, Easy to complete, hard to choose: Investigating llm performance on the proverbit benchmark, in: Proceedings of the Eleventh Italian Conference on Computational Linguistics (CLiC-it 2025), 2025, pp. 722–734.

[8] M. Ernandes, G. Angelini, M. Gori, et al., Webcrow: A web-based system for crossword solving, in: AAAI, 2005, pp. 1412–1417.

[9] E. Wallace, N. Tomlin, A. Xu, K. Yang, E. Pathak, M. Ginsberg, D. Klein, Automated crossword solving, arXiv preprint arXiv:2205.09665 (2022).

[10] J. Rozner, C. Potts, K. Mahowald, Decrypting cryptic crosswords: Semantically complex wordplay puzzles as a target for nlp, Advances in Neural Information Processing Systems 34 (2021) 11409–11421.

[11] A. Sadallah, D. Kotova, E. Kochmar, What makes cryptic crosswords challenging for llms?, in: Proceedings of the 31st International Conference on Computational Linguistics, 2025, pp. 5102–5114.

[12] X. Zhang, J. Cao, C. You, Counting ability of large language models and impact of tokenization, 2024. URL: https://arxiv.org/abs/2410.19730. arXiv:2410.19730.

[13] S. Saha, S. Chakraborty, S. Saha, U. Garain, Language models are crossword solvers, in: 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), 2025, pp. 2074–2090.

[14] N. Shazeer, M. Littman, G. Keim, Solving crossword puzzles as probabilistic constraint satisfaction, Proc. AAAI '99 (1999).

[15] S. Kulshreshtha, O. Kovaleva, N. Shivagunde, A. Rumshisky, Down and across: Introducing crossword-solving as a new NLP benchmark, in: S. Muresan, P. Nakov, A. Villavicencio (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume

1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 2648–2659. URL: https://aclanthology.org/2022.acl-long.189/. doi:10.18653/v1/2022.acl-long.189.

[16] M. L. Ginsberg, Dr. fill: Crosswords and an implemented solver for singly weighted csps, Journal of Artificial Intelligence Research 42 (2011) 851–886.

[17] G. Barlacchi, M. Nicosia, A. Moschitti, Learning to rank answer candidates for automatic resolution of crossword puzzles, in: Proceedings of the Eighteenth Conference on Computational Natural Language Learning, 2014, pp. 39–48.

[18] A. Severyn, M. Nicosia, G. Barlacchi, A. Moschitti, Distributional neural networks for automatic resolution of crossword puzzles, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2015, pp. 199–204.

[19] G. Sarti, M. Nissim, IT5: Text-to-text pretraining for Italian language understanding and generation, in: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, N. Xue (Eds.), Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italia, 2024, pp. 9422–9433. URL: https://aclanthology.org/2024.lrec-main.823.

[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of machine learning research 21 (2020) 1–67.

[21] Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama, A. Vaswani, D. Metzler, Scale efficiently: Insights from pre-training and fine-tuning transformers, arXiv preprint arXiv:2109.10686 (2021).

[22] K. Zeinalipour, T. Iaquinta, A. Zanollo, G. Angelini, L. Rigutini, M. Maggini, M. Gori, Italian crossword generator: Enhancing education through interactive word puzzles (2023).

[23] L. De Moura, N. Bjørner, Z3: An efficient smt solver, in: International conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2008, pp. 337–340.

[24] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, C. Raffel, Byt5: Towards a token-free future with pre-trained byte-to-byte models, Transactions of the Association for Computational Linguistics 10 (2022) 291–306.

[25] Y. Tay, V. Q. Tran, S. Ruder, J. Gupta, H. W. Chung, D. Bahri, Z. Qin, S. Baumgartner, C. Yu, D. Metzler, Charformer: Fast character transformers via gradient-based subword tokenization, arXiv preprint arXiv:2106.12672 (2021).

[26] J. Wang, T. Gangavarapu, J. N. Yan, A. M. Rush, Mambabyte: Token-free selective state space model, arXiv preprint arXiv:2401.13660 (2024).