MINDS at Cruciverb-IT: Solving Italian Crossword Clues with Masked Language Models and Candidate Pooling

Flavio Giobergia

Politecnico di Torino. Turin, Italy

Abstract

Being able to solve crossword puzzles is a task that has engaged humans for more than a century, due to the non-trivial skills it requires – such as logical reasoning, general knowledge, multilingual competence, and wordplay. For these reasons, addressing crossword puzzles is also challenging for language models. The Cruciverb-IT shared task at EVALITA 2026 focuses on this problem for the Italian language. In this paper, we present a methodology for generating answers to crossword clues by leveraging a pre-trained encoder-only transformer model (specifically, BERT) fine-tuned to solve crosswords as a masked language modeling task. We show that a relatively small LM can be efficiently fine-tuned on a large corpus of clue-answer pairs. To mitigate limitations intrinsic to encoder-only architectures, we further propose a strategy that aggregates high-probability predictions to construct a pool of candidate solutions of varying lengths. Our approach achieves competitive performance – e.g., an accuracy@1 of 0.5910 – while remaining computationally efficient compared to approaches relying on substantially larger models.

Keywords

crossword solving, masked language modeling, encoder-only transformers

1. Introduction

Crossword puzzles have long served as a popular and intellectually demanding word game, requiring solvers to integrate linguistic knowledge, cultural references, and various forms of deductive and associative reasoning. For computational systems, these requirements translate into a complex inference task where lexical, semantic, and pragmatic cues must be combined to derive a concise answer – often a single word – from a short, sometimes ambiguous clue. As such, crosswords represent an appealing test scenario for evaluating language models' ability to perform structured reasoning rather than surface-level text matching.

Early work on automatic crossword solving predominantly relied on lexical resources, similarity metrics, or rule-based strategies tailored to specific classes of clues. These approaches typically performed well when clues were literal or closely matched previously encountered phrasing, but their reliance on explicit linguistic patterns limited their capacity to generalize to more creative formulations, figurative uses, or clues involving polysemy and wordplay. With the emergence of neural language models, the task has gained renewed attention, yet even modern LMs continue to encounter difficulties: clues are terse, often require background knowledge, and the answer space is effectively unconstrained, making direct text generation non-trivial – especially for languages with rich morphology, such as Italian.

The Cruciverb-IT [1] shared task at EVALITA 2026 [2] provides a standardized evaluation framework for studying crossword solving in Italian. In this work, we address Task A, which focuses on predicting likely answers for isolated clues when the character length of the target word is known. This formulation isolates the semantic and lexical aspects of crossword solving, without requiring systems to fill complete grids or enforce inter-word constraints. At the same time, the task remains challenging: clues may be ambiguous or culturally grounded, and the correct answers are often short strings that do not directly appear in the clue text.

Our approach builds on the intuition that masked language modeling (MLM) can serve as an effective proxy for crossword clue completion. Instead of relying on retrieval or similarity-based methods, we

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

flavio.giobergia@polito.it (F. Giobergia)

0000-0001-8806-7979 (F. Giobergia)

fine-tune a compact encoder-only transformer model (BERT) to reconstruct missing solution words from their corresponding clues. This modeling choice offers two advantages: (i) the ability to train efficiently on large collections of clue-answer pairs, and (ii) the use of contextualized representations that capture semantic nuances beyond surface similarity. To mitigate the limitations of encoder-only architectures – particularly the challenge of generating variable-length outputs – we introduce a candidate pooling mechanism based on high-probability token sequences, enabling the model to propose multiple plausible solutions of different lengths.

We evaluate our system on the official Cruciverb-IT benchmark and show that this lightweight approach achieves competitive performance while maintaining substantially lower computational requirements than large autoregressive LMs. Our results highlight the potential of encoder-only models, when coupled with targeted fine-tuning and structured candidate generation, to perform non-trivial reasoning over crossword clues in Italian.

2. Methodology

Our system approaches crossword clue answering by reframing it as a MLM task. Given a clue and the character length of the target word, we construct a templated input sequence in which the answer is represented by one or more `[MASK]` tokens. A pre-trained BERT model is then fine-tuned to reconstruct the masked span from the clue context. During inference, the true number of tokens required for the answer is unknown, which is a limitation of encoder-only architectures. To address this, we generate candidate answers by querying the model with multiple possible mask lengths and aggregating predictions across them.

The overall architecture consists of three stages:

1. Input construction and masking,
2. Fine-tuning as a masked language model,
3. Multi-length candidate generation and ranking.

Figure 1 summarizes the proposed pipeline. The following subsections explain each of the steps in more detail.

2.1. Input Construction and Masking

For each clue-answer pair in the training data, we build an input sequence that concatenates the clue text with an explicit indication of the answer length. We adopt the following template:

```
<clue text> (<answer length> lettere) : [MASK] [MASK] ... [MASK]
```

The number of `[MASK]` tokens corresponds to the number of tokens in the ground truth answer, as determined by the tokenizer. Including the character length in parentheses helps the model constrain its predictions to a plausible morphological and lexical space.

At inference time, however, the number of answer tokens is unknown: although the length of the answer, in characters, is known, there is a weak relationship between the length in tokens and the length in characters, as shown in Section 3.1.2. Because encoder-only models do not generate text autoregressively, we must supply a fixed number of `[MASK]` tokens per query. This limitation motivates the candidate-expansion strategy described in Section 2.3.

2.2. Fine-Tuning the Encoder-Only Model

We fine-tuned an Italian BERT model using standard masked language modeling with cross-entropy loss. Only the masked positions contribute to the loss, encouraging the model to reconstruct the answer tokens directly from the clue context. No additional masking is applied beyond the answer span, allowing the model to learn a direct mapping from clue text to its solution.

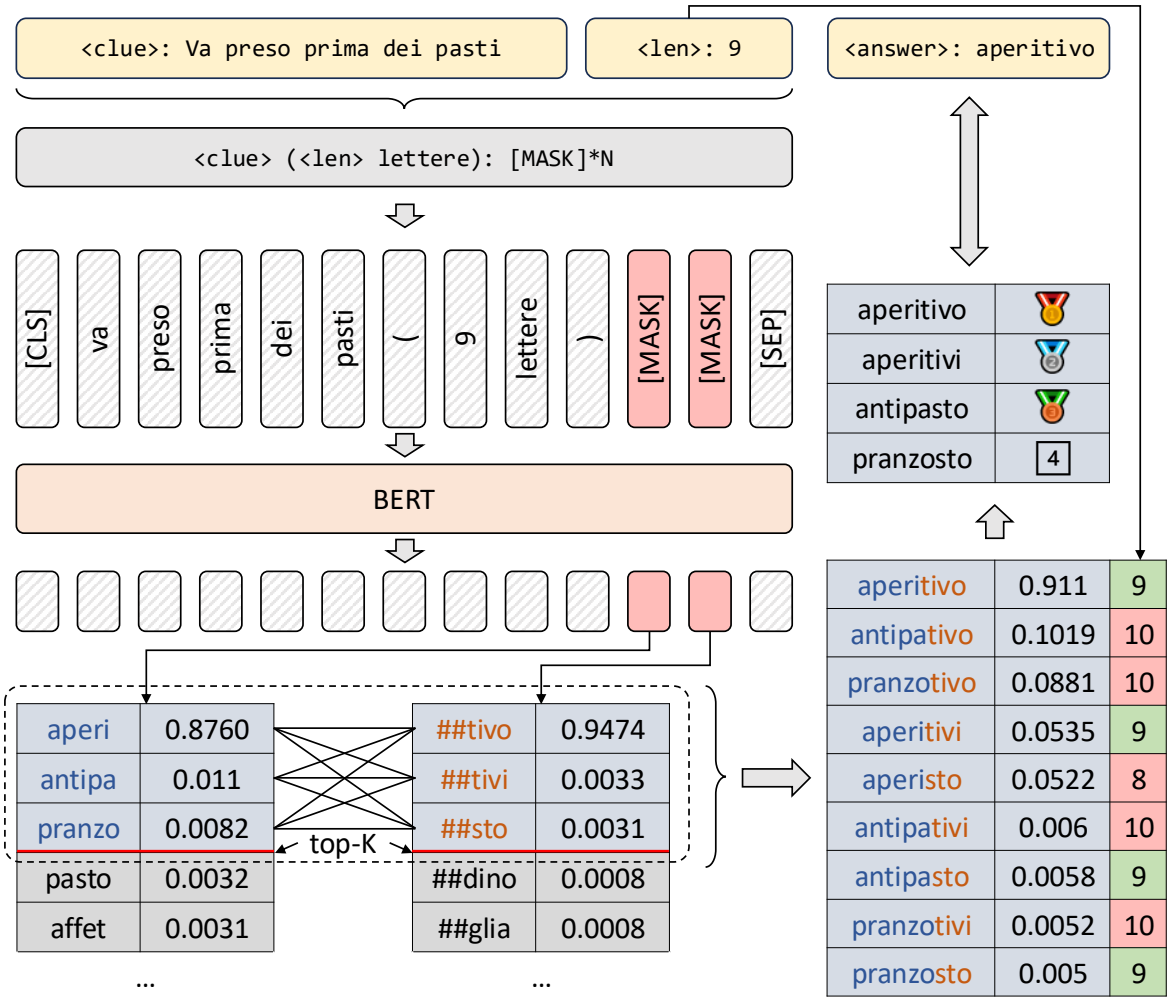

Figure 1: Summary of the proposed methodology.

2.3. Inference with Variable Mask Lengths

Since the correct number of subword tokens for a new answer is unknown, we query the model using a range of hypothesized mask lengths (e.g., 1 to N). For each length m, we create an input sequence matching the structure used during training but containing exactly m `[MASK]` tokens.

For each masked position, the model outputs a probability distribution over the vocabulary. Candidate answers are generated as follows:

1. **Top-K token selection per mask.** For each masked token position, we extract the K most likely predictions (according to model logits).
2. **Token-combination expansion.** For a mask span of length m, we form all possible sequences from the top-K tokens of each position, yielding up to K^m candidates.
3. **Sequence scoring.** Each candidate sequence is assigned a probability score computed as the *geometric mean* of its token probabilities:

$$P(\mathbf{t}) = \left(\prod_{i=1}^{m} p(t_i) \right)^{1/m}.$$

   The adoption of a geometric mean allows normalizing scores across different mask lengths.
4. **Aggregation across lengths.** Candidates produced for different values of m are merged into a single ranked list. It may happen that the same word is produced at multiple mask lengths (e.g.,

aperitivo as *aperi*, *##tivo* (m=2) and *aper*, *##iti*, *##vo* (m=3)). In such cases, only the maximum-score instance is retained.

5. **Pruning invalid answers.** Only the candidate answers that are acceptable are kept. More specifically, candidates with lengths different than then known answer length, and answers containing invalid characters, are pruned – as these cannot be valid answers.
6. **Final selection.** The top 10 highest-scoring candidates are returned.

This procedure enables an encoder-only model – which cannot freely generate variable-length outputs – to approximate generative inference.

3. Experimental results

In this section, we report the results obtained on the answer generation task for Cruciverb-IT (Task 1). First, some relevant details about the dataset are presented, along with a brief overview of the metrics adopted for the evaluation (Section 3.1). Next, the main results are presented for the proposed solution (Section 3.2). The sensitivity to the two main hyperparameters of the proposed solution (N and K) is shown in Section 3.3). Finally, in Section 3.4, we conducted an error analysis to identify the most common patterns that lead to failures.

3.1. Dataset and metrics

The dataset used is the one proposed for the Cruciverb-IT challenge. It is comprised of three splits: training data (374,766 clue-answer pairs), validation data (20,820 clue-answer pairs) and test data (20,821 clue-answer pairs). Models can produce up to 10 ranked answers to a clue. The quality of models is assessed through accuracy@1, accuracy@10 and Mean Reciprocal Rank (MRR).

3.1.1. Data quality

We identify several data quality issues in the dataset:

Missing or incorrect clues/answers. A small number of entries contain missing or erroneous clues or answers, mainly due to issues in data collection. In the training set, 5 clues are reported as *"<Inserire definiz.>"* (*"enter definition"*) and 2 are entirely missing. Manual inspection also reveals a few incorrect answers, such as *"iari"* instead of *"bari"* for *"Il capoluogo della regione pugliese"*, or *"cu"* instead of *"na"* for *"Il sodio in chimica"*. These cases are rare and unlikely to significantly impact overall performance.

Same clue, different answers. Some clues appear multiple times with different answers, a common characteristic of crosswords where answer length or surrounding letters disambiguate the solution. While answer length is provided and helps rule out invalid options, the lack of contextual information makes it impossible to distinguish between multiple valid answers of the same length. In some cases, disambiguation depends on external context, such as *"personaggio del giorno"* clues, whose answers appear elsewhere in the magazine. Clues like *"Male/female name"* exhibit the highest ambiguity, with up to 17 same-length answers. Overall, the training set contains 8,685 ambiguous clue-answer pairs (about 2.3%).

Training and validation/test overlaps. After normalizing clues and answers (e.g., removing accents), we observe overlaps between the training and validation/test sets. We distinguish between *identical* overlaps (same clue and answer) and *misleading* overlaps (same clue and length, different answers). The validation set contains 255 identical and 364 misleading overlaps (about 3%), while the test set contains 269 identical and 395 misleading overlaps (about 3.2%). Identical overlaps may advantage the model, whereas misleading ones may disadvantage it.

Overall, these issues may partially affect the reported results but are expected given the dataset's size. Using multiple evaluation metrics, including accuracy@10, helps mitigate their impact.

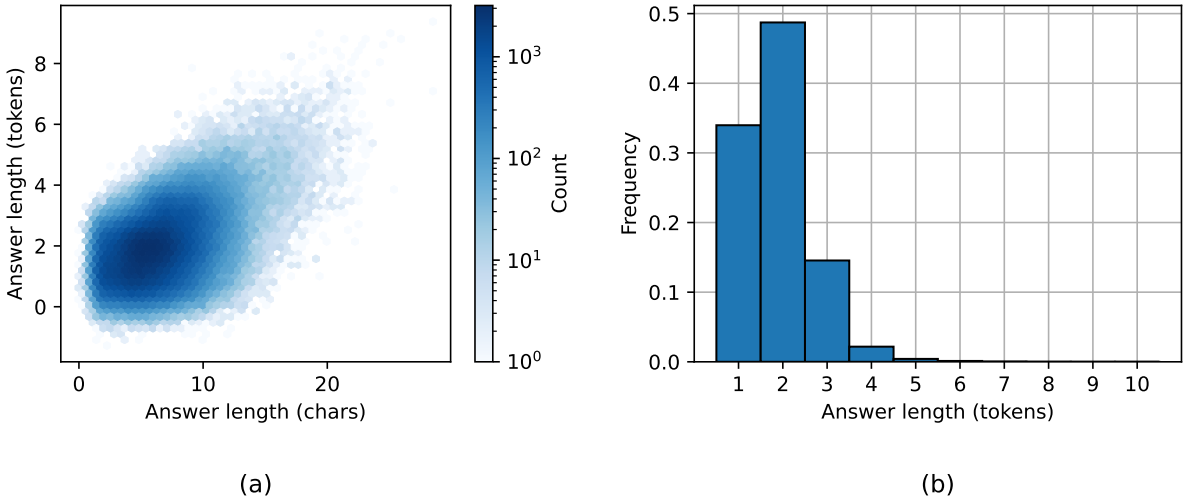

Figure 2: (a). Relationship between answer length in characters, and answer length in tokens. A clear linearl relationship exists, however the variance is large enough to make it difficult to predict the "correct" number of tokens, based only on the length in characters of the answer (the only available information). **(b)**. Distribution of the lengths of the answers (in tokens). The majority of answers can be represented with a limited number of tokens (97% with up to 3 tokens).

3.1.2. Answer lengths

The selected encoder-only architecture has one significant limitation: the number of tokens for the answer need to be specified in advance, as a part of the input (in the form of `[MASK]` tokens). The only information available about the length of the answers is in terms of letters. We qualitatively and quantitatively verify the relationship between the length of answers in tokens and in letters, to decide whether the correct number of tokens can be estimated from the length of the answer in letters.

Figure 2 (a) shows the scatter plot of lengths of answers in letters and tokens, for the training set. A subtle relationship can be observed. This relationship can be quantified with a Pearson correlation of approximately 0.51 (for the tokenizer associated with the selected version of BERT). This suggests that, although there indeed is a relationship between the two lengths, estimating the length in tokens from the length in letters may not be accurate enough.

In addition, 2 (b) shows the distribution of lengths, in tokens, for the answers in the training set. We note that 97% of the answers are composed of 1, 2, or 3 tokens. Given this limited range, we used masks of those lengths (i.e., N = 3) in the proposed methodology. In Section 3.3, we consider different values for N.

3.2. Main results

We report the main results obtained in Table 1. The reported performance has been obtained by training various models (as listed in the table) on the training and validation sets, and reporting the results on the test set.

We selected various BERT-based architectures, namely BERT base [3], RoBERTa [4], as well as two versions that have been pre-trained on the Italian language [5][6].

The baseline reported, presented by the organizers of the challenge, uses BM25 [7] to rank the most similar entries in the training set. The other reported solutions are the ones obtained by the other participants (the runs are unnamed as the information on the teams was not available).

The proposed solution achieves competitive results, especially considering the accuracy@1 and the MRR. In other words, if the model "knows" the solution, it ranks it in the first few positions. This is consistent with the expected behavior, since the highest probability answer is the one that has highest logits for the masked tokens. However, the gap between accuracy@1 and accuracy@10 highlights a

Table 1
Main results of the Cruciverb-IT challenge, Subtask A. For each solution, accuracy@1, accuracy@10 and Mean Reciprocal Rank are reported. For all metrics, higher is better. The named runs belong either to the proposed system, or to the baseline. All 'XXX' entries correspond to submissions made by other participants.

Runs	accuracy@1	accuracy@10	MRR
XXX	0.69	0.83	0.72
XXX	0.58	0.75	0.63
BERT base (Italian)	0.59	0.71	0.62
XXX	0.57	0.75	0.62
XXX	0.55	0.72	0.60
XXX	0.54	0.73	0.60
XXX	0.51	0.73	0.57
RoBERTa base (Italian)	0.5	0.64	0.54
BERT large	0.49	0.61	0.53
XXX	0.47	0.67	0.53
XXX	0.46	0.69	0.52
BERT base	0.47	0.59	0.51
XXX	0.43	0.59	0.47
Baseline	0.40	0.62	0.46
XXX	0.36	0.54	0.41

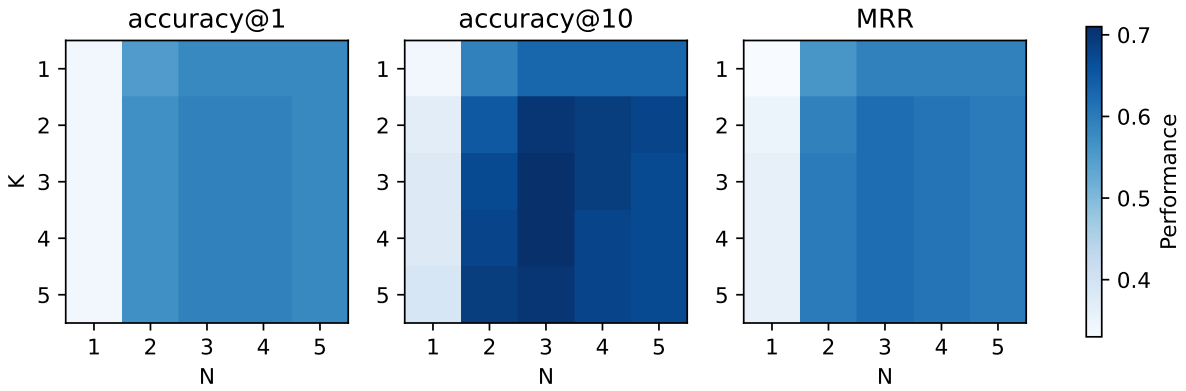

Figure 3: Performance, in terms of accuracy@1, accuracy@10 and MRR, as a function of the K and N hyperparameters.

limitation of the model: there are cases where the correct answer *could* be known (since it is in the top-10), but the model fails to rank it first. The large MRR value suggests that solutions are placed, on average, in the first few positions. However, since the model's proabilities used are the same ones as the generating model, it is unlikely for a different answer (other than the most likely one) to be considered valid. Alternative approaches could use a different, specifically-tuned and lightweight language model (e.g., fastText [8], a fast models that supports out-of-vocabulary words [9]) to re-rank the candidates. This could allow, for instance, to reduce the probability assigned to non-existing words.

3.3. Sensitivity to hyperparameters

In this section, we focus on the sensitivity of the proposed pipeline to the two hyperparameters N (maximum number of mask tokens considered in the template) and K (when considering the K tokens with highest logits in the answer generation phase). We consider a range of values for N from 1 to 5, as this range of values covers the vast majority of answers, as previously discussed. For K, we consider values from 1 to 5 for, as we observed that considering additional values started including very low-probability tails. We report the performance as a function of K and N, according to the three metrics considered this far, in Figure 3.

The results highlight some interesting aspects, which we discuss below.

In terms of N, a value of 1 is an overall poor choice: this is expected, since the majority of answers can only be represented with 2 or more tokens (as shown in Figure 2). Indeed, using $N \geq 2$ produces more stable results.

Regarding K, various considerations can be made. First, we point out that $K = 1$ selects the highest-probability subword for each token. In other words, a single candidate answer (the "most likely" answer) is produced. This can be expected to perform poorly in terms of accuracy@10 and MRR (since both benefit from having multiple candidates being generated). We confirm this to be the case. However, this solution also has lower performance for the accuracy@1: the behavior is counterintuitive, since $K = 1$ should already provide the highest-probability answer. However, we note that the length-based pruning step may discard the only (highest probability) generated candidate, and leave the answer blank. As a consequence, having multiple candidates provides a slight boost in accuracy@1 as well.

The plots show how a choice of $N = 3$ and $K = \{3, 4\}$ produces the most promising results. We select $N = 3$ and $K = 3$, as this configuration introduces the least overhead (i.e., fewer candidates being generated and processed).

3.4. Error analysis

To better understand the failure modes of the model, we additionally present an error analysis session. We apply an existing technique, DivExplorer [10] to find "subgroups" of clues for which the proposed model systematically underperforms. We describe each clue as a bag of words, plus an additional "tag" defining the length of the expected answer.

In this way, we can extract frequent subgroups (in this case, using a support 0.001, i.e. all subgroups containing at least 0.1% of the points in the test set) and compute the performance of the model on these subgroups (in terms of accuracy@1, in this case). Then, we can extract the subgroups for which the model underperforms w.r.t. the global performance. We carry out these operations with the efficient implementation proposed in [11].

We report the 25 most diverging subgroups in Table 2. We observe some interesting families of subgroups, as highlighted in the table. Namely:

- **Long answers.** Subgroups characterized by answer lengths from 10 to 17 letters all appear in the top positions. This is intuitively a reasonable outcome: longer answers are generally more difficult, often involving multiple words (e.g., full names, or concatenations of words), leaving more room for errors.
- **General knowledge.** Another group of entries for which the model generally outperforms is the one about general knowledge. The pretrained model does possess some general knowledge, and some additional information is included during the fine-tuning phase. However, it still fails to answer many clues that rely on historical facts of general knowledge. We include "padre" (father) in this subgroup as it is typically used, both literally and figuratively (to indicate the inventor of something) in a way that still requires general knowledge to address.
- **Definitions.** Another category includes definitions: this requires knowing synonyms or being able to understand descriptions. Once again, the model does have partial knowledge of these definitions (and indeed, most of these subgroups are in the "better performing" part of the top-25). However, the model does lack some knowledge required to properly address some of the clues.
- **Play of words.** Surprisingly, there are many "play of words" clues that the model cannot solve. We argue that this is counterintuitive, as these kinds of clues (e.g., *"Il centro delle Langhe"*, or *"In the middle of Langhe"*, follow fairly simple patterns that should be easily detected by ML models. We assume, although further investigation would be required, that one of the main reasons for these kinds of mistakes lies in the fact that these models work at a token level, rather than at the character level: this has been shown in literature to cause problems when addressing character-level tasks (e.g., in the "strawberry test") [12]. Of the various categories of errors, we believe that this is the one that can be more easily mitigated: for instance, a smaller, character-level model can be introduced to addresses these types of clues).

Table 2
Most diverging subgroups in the test set. Row colors identify four groups of patterns, namely long answers, general knowledge, definitions, and play of words. For each subgroup, the accuracy@1 for that subgroup, the performance gap with the global accuracy@1 (0.5910), and the number of hits and misses (correctly and incorrectly predicted clues) are reported.

Subgroup	accuracy@1	$\Delta_{a@1}$	# hits	# misses
<len=17>	0.0000	-0.5910	0	28
<len=15>	0.0278	-0.5632	1	35
<len=14>	0.1014	-0.4895	7	62
<len=16>	0.1351	-0.4559	5	32
<len=13>	0.1845	-0.4065	19	84
modo	0.2683	-0.3227	11	30
esserlo	0.2857	-0.3053	6	15
<len=12>	0.2869	-0.3041	70	174
far	0.2895	-0.3015	11	27
<len=11>	0.3326	-0.2584	146	293
personaggio	0.3333	-0.2577	12	24
storico	0.3333	-0.2577	10	20
protagonista	0.3333	-0.2577	7	14
italiano	0.3429	-0.2481	12	23
<len=10>	0.3592	-0.2318	315	562
lavora	0.3600	-0.2310	9	16
tedesco	0.3636	-0.2274	8	14
sistema	0.3636	-0.2274	8	14
piccola	0.3636	-0.2274	8	14
centro	0.3696	-0.2214	34	58
<len=2>, centro	0.3778	-0.2132	17	28
padre	0.3793	-0.2117	11	18
anni	0.3793	-0.2117	11	18
<len=2>, fondo	0.3810	-0.2100	8	13
parola	0.3810	-0.2100	8	13

4. Conclusions

We presented an approach to Italian crossword clue solving based on fine-tuning an encoder-only transformer as a masked language model, together with a candidate generation phase to handle variable-length answers. We overcome one of the limitations of encoder-only models by aggregating high-probability token sequences across multiple mask lengths. Experimental results show that this lightweight approach achieves competitive performance, particularly in terms of accuracy@1 and MRR, despite relying on significantly smaller models than autoregressive alternatives. Our analysis highlights both the strengths of MLM for lexical inference and its limitations in handling long answers and character-level wordplay. To address these limitations, future work could explore hybrid architectures that work at a character-level, as well as retrieval-based components.

Acknowledgments

Thanks to the developers of ACM consolidated LaTeX styles https://github.com/borisveytsman/acmart and to the developers of Elsevier updated LaTeX templates https://www.ctan.org/tex-archive/macros/latex/contrib/els-cas-templates.

Declaration on Generative AI

Either:
The author(s) have not employed any Generative AI tools.

Or (by using the activity taxonomy in ceur-ws.org/genai-tax.html):
During the preparation of this work, the author(s) used X-GPT-4 and Gramby in order to: Grammar and spelling check. Further, the author(s) used X-AI-IMG for figures 3 and 4 in order to: Generate images. 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] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).

[5] B. Staatsbibliothek, S. Schweter, bert-base-italian-cased (revision 843e404), 2025. URL: https://huggingface.co/dbmdz/bert-base-italian-cased. doi:10.57967/hf/5850.

[6] roberta-base-italian, 2025. URL: https://huggingface.co/osiria/roberta-base-italian.

[7] S. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, volume 4, Now Publishers Inc, 2009.

[8] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin, Advances in pre-training distributed word representations, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[9] C. Savelli, F. Giobergia, Enhancing cross-lingual word embeddings: Aligned subword vectors for out-of-vocabulary terms in fasttext, in: 2024 IEEE 18th International Conference on Application of Information and Communication Technologies (AICT), IEEE, 2024, pp. 1–6.

[10] E. Pastor, L. De Alfaro, E. Baralis, Looking for trouble: Analyzing classifier behavior via pattern divergence, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 1400–1412.

[11] F. Giobergia, E. Pastor, L. de Alfaro, E. Baralis, Detecting interpretable subgroup drifts, in: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1, KDD '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 366–377. URL: https://doi.org/10.1145/3690624.3709259. doi:10.1145/3690624.3709259.

[12] N. Xu, X. Ma, Llm the genius paradox: A linguistic and math expert's struggle with simple word-based counting problems, 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. 3344–3370.