CrosswordLLM: A Benchmark Framework for Evaluating Large Language Models on Linguistic Reasoning in Crossword Puzzles

Alex Frugé

May 2, 2025

Abstract

This paper introduces CrosswordLLM, a benchmark framework designed to evaluate the performance of Large Language Models (LLMs) in solving crossword puzzles, a task requiring complex linguistic reasoning. Using the New York Times Crossword Clues dataset, the study systematically tests models from the Pythia suite (ranging from 14M to 1.3B parameters) to analyze the impact of model size, architectural enhancements, and prompting strategies such as Chain-of-Thought (CoT) and contextual embeddings. Results reveal that while larger models exhibit modest improvements in accuracy and length matching, overall performance remains low, highlighting the challenges LLMs face in tasks demanding structured outputs and nuanced reasoning. The findings underscore the limitations of current LLMs in crossword solving and suggest directions for future work, including targeted training and integration of advanced reasoning techniques.

Introduction

Large Language Models (LLMs) have demonstrated remarkable proficiency in natural language understanding and generation. However, their ability to solve complex linguistic tasks, such as crosswords, remains an open area of investigation. This project aims to design an evaluation pipeline to test how well different LLMs can solve crossword clues. By analyzing performance variations across model sizes and architectures, I seek to understand whether larger models inherently perform better, or if architectural differences contribute more significantly to success. Additionally, I will explore how contextual information, prompt engineering, and explicit length requirements for answers may affect model performance. This research will not only offer insights into the cognitive capabilities of LLMs but also contribute to broader AI evaluations in natural language processing (NLP).

Literature Review

Foundational Work

The sources within the Foundational Work section mainly go discuss how LLMs scale better with more compute, which was one of the attributes I wanted to verify in my crossword case.

- Brown et al. (2020) [1] introduced GPT-3, demonstrating that large-scale LLMs can perform a variety of tasks with minimal examples, a technique known as few-shot learning. This work established the potential of LLMs to generalize across tasks without extensive fine-tuning.
- Chowdhery et al. (2022) [2] presented PaLM, a 540-billion-parameter model trained using the Pathways system. PaLM achieved state-of-the-art results on numerous benchmarks, highlighting the benefits of scaling model size and training data for improved performance.

Enhancing Few-Shot Learning and Prompting Techniques

This section highlights methods for improving language model performance with limited data through better prompt design and reasoning strategies.

- Gao et al. (2020) [3] proposed LM-BFF, a method combining prompt-based fine-tuning with automated prompt generation to improve few-shot learning in smaller models. This approach is particularly relevant for scenarios with limited annotated data, such as specialized crossword clues.
- Wei et al. (2022) [4] introduced Chain-of-Thought (CoT) prompting, where models generate intermediate reasoning steps before arriving at an answer. CoT prompting significantly improved performance on complex reasoning tasks, suggesting its applicability to the nuanced reasoning required in crossword solving. I ended up using Chain of Thought prompting as one of my enhancements to the existing models.
- Zhou et al. (2023) [5] developed Automatic Prompt Engineer (APE), a system that automatically generates and selects effective prompts.

APE outperformed human-crafted prompts on various tasks, indicating the potential for automated prompt optimization in crossword applications.

Previous Applications to Crossword Solving

Saha et al. (2024) [6] demonstrated that current LLMs exhibit significant competence in solving cryptic crossword clues, outperforming previous state-of-the-art results by a factor of 2–3. They developed a search algorithm enabling LLMs to solve full crossword grids, achieving 93% accuracy on New York Times puzzles. This work underscores the feasibility of applying LLMs to complex language tasks like crosswords

Emerging Techniques and Considerations

Pattern-Aware Chain-of-Thought Prompting (Zhang et al., 2024) [7] emphasizes the importance of diverse reasoning patterns in prompts to enhance model generalization and robustness. This approach could further improve LLM performance on varied crossword clue types.

Future Direction

Integrating these advancements suggests a multifaceted approach to developing an LLM-based crossword-solving suite:

- Model Selection: Utilize large-scale models like PaLM for their superior reasoning capabilities.
- Few-Shot Learning: Implement LM-BFF techniques to fine-tune models with limited crossword-specific data.
- Reasoning Enhancement: Apply CoT and pattern-aware prompting to improve the model's ability to handle complex clues.

By building upon these foundational and contemporary works, the proposed suite aims to advance the capability of LLMs in solving crossword puzzles with high accuracy and efficiency.

Methodology

Data Sourcing & Explanation

The data for this project was pulled from Darin Hawley's New York Times Crossword Clues & Answers 1993-2021 dataset. (Hawley, 2021) [8] This dataset contains 781,573 entries, each of which come with the date the puzzle containing the clue was published, the clue itself, and the answer.

The NYT dataset is being used as our training data, as it contains a good spread of clue variety, including some pop-culture related clues, some wordplay, and a wide variety of difficulty in the clues themselves. It also contains a good number of clues involving multiple words in a clue being concatenated into one continuous word (see Table 1 for some examples of this). I'll also use the NYT dataset for a testing suite that tests the selected models on more conventional clues.

Libraries and Tools Used

The core libraries used within this package are **PyTorch 2.0** (training original models), **TransformerLens** (loading original models), **HuggingFace Transformers** (loading tokenizers), as well as **Pandas** and **Scikit-learn**.
The system leverages the **Pythia** suite of autoregressive language models (Biderman et al., 2023) [9] as its primary base architecture due to their transparency, reproducibility, and controlled scaling properties. Pythia models provide a standardized framework for experimentation, with versions ranging from 70M to 12B parameters, allowing systematic comparisons across model sizes. For the sake of the models being usable on my computer, the largest model I tested with contained 1.3 billion parameters.

Training Process

The training process follows a structured pipeline designed to optimize the model's ability to generate accurate crossword answers while incorporating task-specific enhancements. Below is a high-level overview of the training workflow.

The system ingests a structured dataset of (clue, answer, length) tuples, derived from the NYT Crossword Clue Dataset. The clues are then standardized (removing metadata like enumeration), and answers are normalized to uppercase alphabetic strings. Answer lengths are explicitly annotated, either as single integers (for single-word answers) or comma-separated values (for multi-word answers). Data is partitioned into training (80%) and evaluation (20%) sets, ensuring no overlap.

Clues are formatted into instructional prompts (e.g., "Solve the crossword clue: [CLUE] ([LENGTH]). Answer:") to guide the model's generation. Multiple prompting strategies (basic, detailed, chain-of-thought) are explored to improve task alignment. Inputs are tokenized using the base model's pretrained tokenizer, with padding/truncation to a fixed sequence length.

Once all of the data is brought in and cleaned up, the user specifies which model they want to use as a base, as well as which enhancement they want to add to that base via a CLI (e.g. main.py –model EleutherAI/pythia-70m –enhancement length_aware would train the pythia-70m model with the length aware enhancement added on).

The model is trained via causal language modeling, where it predicts the answer tokens autoregressively. The loss is computed only on answer tokens (excluding the prompt). Depending on which enhancements the user specifies, there could be intervention here by said enhancements.

The training loop itself uses the AdamW optimizer with a learning rate of $5E-05$, batch size of 32, and gradient clipping, as well as dropout (where applicable) and early stopping to prevent overfitting.

Enhancements

The base autoregressive language model is augmented with several task-specific enhancements designed to improve its crossword-solving capabilities. These modifications target key challenges in clue-answer generation, including length conditioning, semantic grounding, and controlled decoding.

Contextual Embedding Fusion LMs may lack deep semantic understanding of clues (e.g., parsing wordplay like "Composer of 'The Planets' (5)" → "HOLST").

The solution that I came up with is a hybrid architecture combining a **frozen sentence embedder** and **gated fusion**.

The frozen sentence embedder is derived from a pretrained model (e.g., all-MiniLM-L6-v2) encodes clues into dense vectors, capturing synonym relationships ("composer" → "musician") and entity linkages ("The Planets" → "Gustav Holst").

The LM's hidden states are combined with projected embeddings via:

$$\mathbf{h}_{\text{fused}} = \sigma\left(\mathbf{W}_g\left[\mathbf{h}_{\text{LM}}; \mathbf{h}_{\text{emb}}\right]\right) \odot \mathbf{h}_{\text{LM}} + \left(1 - \sigma\left(\mathbf{W}_g\left[\mathbf{h}_{\text{LM}}; \mathbf{h}_{\text{emb}}\right]\right)\right) \odot \mathbf{h}_{\text{emb}}$$

where $\mathbf{W}_g$ is a learned gate matrix.

Prompt Engineering Strategies The final problem I considered was how clues can require different reasoning modes (definition, wordplay, trivia). I explored using different formatted prompts to guide the LM's behavior. In the end, I came up with three different prompt formats which each have a different target (detailed in Table 2).

Testing Process

The evaluation framework rigorously assesses model performance across multiple dimensions, combining automated metrics with human analysis to ensure comprehensive validation of crossword-solving capabilities.

Results

Base Models

The first test I ran was running all of the models I selected (pythia-14m through pythia-1.3b) with no enhancements, with the goal of verifying that models increase performance as their size increases.

The results in table 4 provide insight into how the performance of models in the Pythia suite scales with size when solving a task based on crossword clues from the New York Times dataset. Here's a summary and interpretation of the findings:

As shown by the results, there is a clear trend showing that both Accuracy and Length Match Accuracy generally improve as model size increases. For example, EMA starts at 0.0000 for the smallest models (pythia-14m, 31m, and 70m), and then slowly begins to increase, reaching a max at 0.0270 for pythia-1.3b. LMA follows a similar upward trend, with the pythia-31m surprisingly achieving the highest score here (0.2580), though results fluctuate slightly across models. My guess for this is that the model is getting lucky with the correct length once the model is large enough, since the LMAs hover around 0.2 once pythia-31m and larger are tested.

This suggests that model scale correlates positively with performance, though the relationship is not strictly linear. Small models struggle significantly, likely due to a lack of capacity to represent the complex relationships required to solve crossword clues—a task that typically relies on semantic associations, cultural knowledge, and language nuance.

The fact that EMA remains low even for the larger models (only 2.7% for pythia-1.3b) indicates that this is a very difficult task, likely involving not just language understanding but also lateral thinking, wordplay, and perhaps common-sense reasoning. The low scores suggest that these models may not yet be well-equipped for this form of symbolic reasoning or require more targeted training.

The Length Match Accuracy is a more forgiving metric, it simply measures whether the model's output matches the length of the correct answer. This improves more noticeably with model size, showing that larger models are better at mimicking superficial features of the answer even if they can't yet solve the clue. The pythia-31m's unusually high score on this metric may be an outlier or a quirk of training dynamics.

Contextual Embeddings

My first enhancement I tested was adding the contextual embeddings to all of the previous models aside from pythia-1.3b.

All models (pythia-14m to pythia-410m) achieve 0.0000 EMA, meaning they fail to produce correct answers despite embedding enhancements (see Table 5). This matches the zero-accuracy trend observed in earlier tests (with and without CoT), reinforcing the task's difficulty.

There are some interesting behaviors from this model when you look at some of the outputted answers (Table 6). It outputs placeholder strings ("ANS", "ANSWER") for clues where the length matches (e.g., "Get hitched to (3)" → "ANS" [3 letters]). This explains its non-zero LMA. This likely means that the model struggles with wordplay (e.g., "Hookups for a camera?" → "SEXTAPES" is correct, but the output "ANSWER" suggests it defaulted to a generic response).

Chain of Thought Testing

The final test I tried was using Chain of Thought prompts on some of the smaller models.

The performance shown in Table 7, which tests models with Chain of Thought (CoT) Prompt enhancement, contrasts with the initial test results. All models (pythia-14m, 31m, 70m, 160m) achieve 0.0000 Accuracy, indicating they fail to produce correct answers even with CoT prompts. This is worse than the initial test, where larger models (e.g., pythia-1.3b) achieved modest accuracy (0.0270).

Chain of Thought prompting does not improve performance for these smaller Pythia models on the NYT crossword task. The zero Accuracy and depressed LMA scores indicate that CoT may overcomplicate the task for models with limited parameters, disrupting even superficial metrics like length matching. This aligns with the initial finding that larger models perform better, but suggests CoT is ineffective without sufficient scale. Further investigation could explore whether CoT benefits emerge with larger models (e.g., pythia-1.3b) or alternative prompt designs.

Conclusion

Overall, the results of this experiment suggest that while increasing model size generally improves performance on crossword clue-solving tasks, the overall accuracy remains low—even for the largest models tested—highlighting the difficulty of the task and the limitations of current LLMs in this domain. In addition, both attempts to enhance the pythia models with embedding strategies and chain of thought prompting did not yield meaningful improvements; as shown in Tables 6 and 7, the generated answers were often either incorrect, generic (e.g., "ANSWER"), or completely absent. This suggests that embedding enhancements or chain of thought prompting alone are insufficient to address the deeper reasoning and cultural knowledge needed for crossword solving. What worked moderately well was the model's ability to match answer lengths more closely as size increased, indicating some understanding of structural patterns. However, what didn't work was the models' ability to generate semantically and contextually accurate responses, even with enhancements. This reveals a gap between pattern recognition and true language understanding, suggesting future work should explore more targeted training, domain adaptation, or multi-step reasoning architectures to bridge that divide.

For future work on this project, I'd love to be able to train on GPUs in order to use larger models, as I think one of the bottlenecks that any LLM will run into during training is model size. This would also allow me to use newer, better models that perform better with less parameters. Another nice extension I would like to make is the integration of other LLM API's such as OpenAI's, in order to test cutting edge models on the crossword clue dataset. The way that the codebase is currently set up would make this a reasonable addition, needing minimal changes.

I think the most important thing to take away from this project is that LLMs may struggle with tasks whose results need to be highly structured, given the way that their output is generated. It's hard to LLMs to check if the tokens they're generating are the proper "length" in order to answer a crossword clue, and this is a problem that can extend to other, less trivial tasks.

References & AI Use

References

[1] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *arXiv preprint arXiv:2005.14165* (2020). URL: `https://arxiv.org/abs/2005.14165`.

[2] Aakanksha Chowdhery et al. "PaLM: Scaling Language Modeling with Pathways". In: *arXiv preprint arXiv:2204.02311* (2022). URL: `https://arxiv.org/abs/2204.02311`.

[3] Tianyu Gao, Adam Fisch, and Danqi Chen. "Making Pre-trained Language Models Better Few-shot Learners". In: *arXiv preprint arXiv:2012.15723* (2020). URL: `https://arxiv.org/abs/2012.15723`.

[4] Jason Wei et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 24824–24837. URL: `https://arxiv.org/abs/2201.11903`.

[5] Yongchao Zhou et al. "Large Language Models Are Human-Level Prompt Engineers". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2023. URL: `https://arxiv.org/abs/2211.01910`.

[6] Soumadeep Saha et al. "Language Models are Crossword Solvers". In: *arXiv preprint arXiv:2406.09043* (2024). URL: `https://arxiv.org/abs/2406.09043`.

[7] Yufeng Zhang et al. "Pattern-Aware Chain-of-Thought Prompting in Large Language Models". In: *arXiv preprint arXiv:2404.14812* (2024). URL: `https://arxiv.org/abs/2404.14812`.

[8] Darin Hawley. *New York Times Crossword Clues Answers 1993-2021*. Nov. 2021. URL: `https://www.kaggle.com/datasets/darinhawley/new-york-times-crossword-clues-answers-19932021/data`.

[9] Stella Biderman et al. "Pythia: A suite for analyzing large language models across training and scaling". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 2397–2430.

https://claude.ai/share/33fa78db-2dc8-4fef-a80b-baa16c511aba

This Claude conversation gave me inspiration for a few enhancements, as well as some starter code for those enhancements.

I also used GPT 4o for help with writing this paper, and explaining some of the harder concepts (gate matrices mainly).

Appendix

Table 1: *Example clues from NYT Crossword Clues.*

Answer	Clue
EATAT	Irritate
ALQAEDA	War on terror target
ASGOODASGOLD	100% reliable

Table 2: *Prompt Engineering Strategies for Crossword Solving*

Strategy	Example Prompt Structure	Use Case
Basic	`"Clue: [CLUE] ([LENGTH]). Answer:"`	Simple, definitional clues
Detailed	`"Task: Solve this clue. Clue: '[CLUE]'. Answer must be [LENGTH] letters, no spaces. Answer:"`	Complex clues
Chain-of-Thought	`"Analyze the clue '[CLUE]'. Consider: 1) Definitions 2) Wordplay. Then give a [LENGTH]-letter answer:"`	Wordplay/metaphorical clues

Table 3: *Comprehensive Evaluation Metrics for Crossword-Solving Models*

Metric Category	Description	Interpretation
Exact Match Accuracy (EMA)	% of answers which are correct	Higher is better
Length-Adjusted Accuracy (LAA)	% of answers with correct length	Meets crossword grid limits

Note. All metrics computed on held-out test set with $n = 1000$ samples. Embedding similarity uses `all-MiniLM-L6-v2`.

Table 4: *First run across Pythia Suite using NYT Data (n=12,800, batch size=32, epochs=5)*

Model	EMA	LMA
pythia-14m	0.0000	0.0320
pythia-31m	0.0000	0.2580
pythia-70m	0.0000	0.1210
pythia-160m	0.0030	0.2150
pythia-410m	0.0230	0.2400
pythia-1.3b	0.0270	0.1970

Table 5: *Testing models with Embedding enhancement using NYT Data (n=12,800, batch size=32, epochs=5)*

Model	Accuracy	Length Match Accuracy
pythia-14m	0.0000	0.0000
pythia-31m	0.0000	0.0000
pythia-70m	0.0000	0.1210
pythia-160m	0.0000	0.2150
pythia-410m	0.0000	0.0090

Table 6: *Example outputs from pythia-70m with Embedding enhancement using NYT Data (n=12,800, batch size=32, epochs=5)*

Clue	Actual	Generated Answer
Carrie in "Sex and the City" (8)	BRADSHAW	*no answer*
Get hitched to (3)	WED	ANS
Attempt to block (6)	OPPOSE	ANSWER
Hookups for a camera? (8)	SEXTAPES	ANSWER

Table 7: *Testing models with Chain of Thought Prompt enhancement using NYT Data (n=12,800, batch size=32, epochs=2)*

Model	Accuracy	Length Match Accuracy
pythia-14m	0.0000	0.1410
pythia-31m	0.0000	0.0920
pythia-70m	0.0000	0.1040
pythia-160m	0.0000	0.2150