Taking a Shot at Solving Crossword Puzzles

Selina He sh4337@nyu.edu
Brian Pennisi bp2221@nyu.edu
Jiawen Wu jw1562@nyu.edu
Ted Xie tx607@nyu.edu

Abstract

There exists little prior research for solving American-style crosswords using transformer-based language models. We explore how these increasingly large language models perform on this task in general and with prompting. We ran experiments by coming up with a list of different ways to prompt the model in both zero- and few-shot settings. We show that we can improve upon the performance of existing fine-tuned and retrieval transformers using InstructGPT with prompting, achieving 38% Exact Match accuracy compared to 24% for the next best model. Our best approach achieves the highest known accuracy of transformer-based language models on these puzzles.

1 Introduction

Crosswords are word puzzles consisting of clues and answers used to fill in a square or rectangular grid. Crossword clues and answers can take many forms. In American-style crosswords, such as the New York Times crossword, some clues are simple definitions of the answers, while others can be fill-in-the-blank or involve wordplay or puns. Similarly, the answer can be a single word, a phrase, a proper noun, an abbreviation, or a modern colloquialism. As such, American crossword puzzles are an open-domain question answering, constraint satisfaction problem that typically require a firm grasp of the English language to solve. In order to reach the correct answer, humans utilize their knowledge of synonyms and linguistic semantics as well as trivia from a wide range of topics. Using modern large language models to solve crossword puzzles can help evaluate the limits of current language models while providing humans with a useful tool to solve these multi-topic puzzles.

There exists little previous literature of attempts to solve crosswords puzzles using transformer-based language models. We test InstructGPT's (Brown et al., 2020), (Ouyang et al., 2022) ability to return the correct answer given a standard crossword clue through multiple combinations of prompting the model. We start with a more general prompt, giving only information typically known at the beginning of trying to solve a crossword (ie. zero letter-based constraints), and we continue to provide more specific prompts with information known only when the puzzle is more filled out. Since previous GPT-3 literature shows promising results for zero and few-shot learning on a broad range of tasks (Radford et al., 2019) (Brown et al., 2020), our experiments mirror similar learning settings.

2 Related Works

Solving a crossword puzzle has been a problem tackled before in an attempt to help create more flexible or creative language models. We saw that Snippet Reranking and Similar Clue Retrieval are methods that can help crossword solvers improve (Barlacchi et al., 2014). Our work will not focus on reranking, however we will borrow the Barlacchi et al. paper's brief history of crossword solvers.

At its core, the problem is very similar to trying to create artificial intelligence, such as DeepQA, that can compete in the Jeopardy! quiz show (Ferrucci et al., 2010); both tasks require an understanding of Question Answer (QA) models. However, answering trivia questions tend to be more straightforward information retrieval for humans. Crosswords puzzle clues not only include trivia but can also contain wordplay, which can be difficult for a language model to interpret. Having machine learning models able to untangle play-on-words and brain twisters can lead us to a more comprehensive understanding of general language models.

One of the earliest systems able to automatically solve crossword puzzles was Proverb (Littman et al., 2002). It uses a probabilistic approach and relies on a large bank of data, including information from the internet and encyclopedias. Another approach was Dr. Fill (Ginsberg, 2014), which tackles

the task as a weighted constrained satisfaction problem (CSP). Solving for a CSP is not particularly efficient either and relies on a large bank of data to train on. An approach using Distributional Neural Networks (Severyn et al., 2015) performed well for its time, but the solver used 2 million clue answer pairs to reach a higher accuracy. Recently in 2021, a more fine-tuned Dr. Fill placed first place in the American Crossword Puzzle Tournament, beating the human benchmark.

Even with a well-performing automatic crossword solver, there is not much prior research done on transformer-based architectures in zero- and few-shot settings for American-style crossword puzzles. By leveraging pre-trained transformer-based language models, we hope to create a method that can efficiently solve crosswords while relying on a lesser amount of data. Large scale language models have been shown to greatly improve performance when prompted with several examples, or few-shot learning, especially using OpenAI's InstructGPT language model. The most similar works involve fine tuning T5 for cryptic crosswords, a variation of traditional crossword puzzles. These were not able to beat the human benchmark (Efrat et al., 2021), (Rozner et al., 2021).

3 Methodology

3.1 Data

We sampled 10,000 clue and answer pairs from New York Times Crossword Clues and Answers 1993 - 2021 Kaggle Dataset[1] for our experiments. For each clue and answer pair, we also obtained the number of letters the answer contains. The dataset offer clues that range in length, style and complexity. Examples in Table 1 show clues can consist of a single word or be a complete sentence. In addition, clues can be laid out in special forms such as question format, fill in the blank or in quotations. Clues may simply be the definition or synonym of the answers. However it can also require the solver to draw on references and wordplay. Answers do not contain punctuation but may be multi word terms condensed together such as "allnighter" or "winorlose." The diverse set of crossword clue answer pairs provided a challenging but realistic test set for InstructGPT on crossword solving. It should also be noted that even an omniscient crossword-solving language model will not get every answer exactly correct since the answer may be a verb where the tense depends on the answer to adjacent clues, such as "swim" versus "swam".

Clue	Length	Answer
Action done while saying "Good dog."	3	pat
"___ what?"	3	say
One hell of a writer?	5	dante
Something that may be pulled in college	10	allnighter

Table 1: Examples of clue answer pair from crossword dataset

3.2 InstructGPT

The language model that we utilized and tested for the task of solving crossword puzzles is rooted in OpenAI's GPT-3 (Brown et al., 2020). GPT-3 is an autoregressive natural language model with 175 billion parameters, pre-trained on a variety of text corpora, making it suitable for answering trivia-style crossword clues. It uses a transformer architecture to generate natural language text and can be "fine-tuned" to specific tasks by varying the prompt and using few-shot learning. This way it can be easily evaluated on several versions of prompts to test the model's ability to solve crosswords given various inputs. In particular, we are used a GPT-3 based model called InstructGPT. This model was trained on GPT-3 responses using reinforcement learning from human feedback (Christiano et al., 2017). The technique involves using a reward function which is engineered by predicting which of a set of responses a human labeler would prefer. The result is a model which is "safer, more helpful, and more aligned"[2] than the baseline GPT-3 model.

4 Experiment

We tested performance of InstructGPT on solving crossword puzzles under zero-, one-, and few-shot learning settings with various prompts. Since language models have been shown to be highly sensitive to prompts (Ribeiro et al., 2020), we took care to find prompts which produce the best results. For example, one experiment which suffixed the prompt with ":" reduced the number of multi-word responses by 7.5%. Since solving crosswords requires one to have extensive knowledge of the English language, we utilized InstructGPT's

[1] https://www.kaggle.com/datasets/darinhawley/new-york-times-crossword-clues-answers-19932021

[2] https://openai.com/blog/instruction-following/

latest and most capable model text-davinci-002 for our experiments. The temperature was set to 0 to ensure deterministic responses. We left the max_tokens parameter to its default value since we also wanted to collect information about how well the model is able to follow instructions which constrain the amount of text that should be provided.

Baseline Prompt

The baseline prompt was given under zero-shot learning setting. The prompt reveals the least amount of information possible by providing only the clue and number of letters the answer should be. We appended the standard default QA suffix "Answer:" on to the prompt. Here is an example of a baseline prompt below:

> Solve this crossword puzzle by providing a 8 letter response to the clue.
> Clue: Places on travel advisory lists.
> Answer:

Constrained Prompt

The constrained prompt was also given under the zero-shot setting. This prompt type mimics the common scenario where one doing the crossword knows specific letters of a clue after completing parts of a crossword. A letter is randomly selected to be revealed as an additional clue in the constrained prompt. The general layout and InstructGPT setting is the same to ensure comparable results with baseline prompt. Note that this does not exactly match the environment of a crossword puzzle in two ways. First, the player may receive more than one letter hint at non-random locations determined by the setup of the board. Second, the number of these letter clues ultimately depends on your ability to solve other clues. As such, the prompts we presented to our InstructGPT model can be more difficult than what a normal player can expect to face. Future work should focus on implementing these results in an end-to-end system which plays the game itself, or with user interaction. We show an example of a constrained prompt below:

> Solve this crossword puzzle by providing a 8 letter response to the clue *where the 2nd letter is o*.
> Clue: Places on travel advisory lists.
> Answer:

Few-Shot Prompt

We experimented with providing one or three examples to the language model in both the baseline and constrained setting. In providing examples, we hoped the model will learn the correct format for answers. In addition, we anticipated more examples reveal subtle patterns of the crosswords. The examples shown for few-shot prompt were randomly selected each time. Due to monetary constraints, we were only able to run 5k samples for three shot learning prompts[3].

Evaluation

Performance was evaluated first on Exact Match (**EM**), that is what percentage of the crossword clues received responses that match the answer exactly. Next, we examined the Constraint Match (**CM**), percentage of responses that match the constraints. For baseline prompts, the only constraint is the answer length. Constraint prompt has additional constraint of having a particular letter at a particular spot. We also provide **EM Norm** and **CM Norm**, where punctuation and white space are stripped from the output prior to metric calculations since crosswords answers are continuous string of letters. This measure is useful for understanding how well the model can aid a human user who would presumably be able to read through the superfluous characters.

5 Results and Analysis

We compared our results with that of another paper (Anonymous, 2022) which looked at the performance of various Transformer-based models on the same NYT crosswords. This paper include two models, BART (Lewis et al., 2020a) and T5 (Raffel et al., 2019), which are fine-tuned on the crosswords. It also includes a retrieval-augmented generation (RAG) (Lewis et al., 2020b) model which supplements its encoder with documents from Wikipedia and English dictionaries to improve performance on certain question answering tasks. Note these models are similar but not the same as InstructGPT, which uses only uses fine-tuning via one- and few-shot learning. Also, the comparison models were trained and evaluated on 1993-2018 data whereas our models looked at 1993-2021 data. We utilized the results of their best performing model, RAG wiki, as a baseline since our

[3]Metrics calculated for other prompts with only 5k samples vs 10k samples did not show significant difference in performance

Prompt Type (# samples)	EM	EM Norm	CM	CM Norm
Baseline RAG wiki (72.9K)	24.2	26.0	NaN	NaN
Zero-Shot Baseline Prompt (10K)	21.4	28.8	49.6	63.2
Zero-Shot Constrained Prompt (10K)	33.3	32.1	36.3	44.1
One-Shot Baseline Prompt (10K)	33.0	35.0	67.4	69.6
One-Shot Constrained Prompt (10K)	35.9	37.7	48.3	50.0
Three-Shot Baseline Prompt (5k)	35.8	36.6	66.0	67.8
Three-Shot Constrained Prompt (5k)	37.6	38.9	48.5	50.0

Table 2: Performance of InstructGPT on zero-, one- and three- shot learning with baseline prompts and constrained prompts. EM and CM stands for Exact Match and Constraint Match metric described in Section 4.

zero-shot baseline prompts outperformed the models in most, if not all, categories. Table 2 compares the performance of InstructGPT in zero-, one-, and few-shot learning settings under the baseline and constrained prompts.

InstructGPT was able to perform on par with the baseline RAG wiki model under the simplest setting where the least information is provided, achieving similar EM in the low 20s when given zero-shot baseline prompts.

One-shot learning was effective in demonstrating the crossword answering format to the model. As shown in the Appendix, 39.3% of zero-shot baseline prompt outputs contained punctuation or white space, whereas one- and few-shot outputs had this issue less than 8% of the time. As a result, only under the zero-shot learning setting was the difference between unnormalized and normalized metrics (EM vs EM norm, CM vs CM norm) greater than 5%.

More examples in few-shot learning did not significantly increase Exact Match or Constraint Match accuracy. For both baseline prompts and constrained prompts, the additional two examples provided by three shot prompts led to less than a 3% boost in EM metric. Still, it remains to be seen whether using K=10 or 100 few-shot prompting would further improve performance as it did for GPT-3 (Brown et al., 2020). We did not test this due to budgetary constraints.

Constrained prompts were harder for the model to fulfill all the conditions compared to baseline prompts. We hypothesize this is because for constrained prompts, InstructGPT has to not only provide an answer with the right response length but also match the letters at the correct position. The additional information of what letter is at which spot enables InstructGPT to achieve slightly more EM accuracy than it does under baseline prompt. This is most evident in the zero-shot setting where constrained prompting resulted in 12% higher EM accuracy than baseline prompting. However, the performance gap between baseline prompt and constrained prompt diminished through few-shot learning. Here, it seems the examples provided sufficient context on how to solve the problem.

Interestingly, the constrained prompt resulted in worse CM performance, despite improving EM performance. This is surprising because the constraints should guide the model to follow the rules more often. One reason why this could be the case is that the constraint reduces the domain of possible outcomes for the language model. While this increases the EM by tuning precision, it may put the model in situations where it can't find an answer which matches the full constraint.

6 Conclusion

Our experiments show that InstructGPT's language model outperforms the baseline RAG model based on EM accuracy. Further, aligned with previous research, few-shot learning settings with more constrained prompts yield higher accuracy than our baseline prompt for solving American crossword puzzles. As more information was disclosed to the model, the system was able to learn enough information to generally answer trivia and solve riddles. The baseline results gave a 21% accuracy and increased to 38% in the few-shot constrained setting in our experiments. We observed that the constraining prompts improved the performance for Exact Match but not for Constraint Match. Future work can explore incorporating additional crossword constraints on the language model, such as crossword themes, word tenses, and rebuses.

7 Ethical Considerations

An automated crossword solver can help assess the development of a language models. However, there are few scenarios where the model could be considered harmful. First, the models are easily accessible to people competing in crossword puzzle tournaments. Dr. Fill was able to beat the human benchmark, but the model was not eligible on taking home the prize, since the tournament is a test of human ability. Cheating could become more common.

As we pointed out throughout the paper, the project was designed to test the limits of transformer language models. We demonstrated InstructGPT can already perform much better with zero-shot learning, suggesting the model can be used broadly. It is possible to misuse the model to generate misinformation.

8 Collaboration Statement

We all worked on the project equally. Selina and Jiawen worked on the code for running our evaluation tests. Brian and Ted gathered research on related papers and the background of existing models. Collectively, we all took part in analyzing the results.

Selina He (sh4337)
Brian Pennisi (bp2221)
Jiawen Wu (jw1562)
Ted Xie (tx607)

9 Github

All source code for this project can be found at:
`https://github.com/sh4337/taking_a_shot_at_crosswords`

References

Anonymous. 2022. Get your model puzzled: Introducing crossword-solving as a new nlp benchmark.

Gianni Barlacchi, Massimo Nicosia, and Alessandro Moschitti. 2014. Learning to rank answer candidates for automatic resolution of crossword puzzles. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 39–48, Ann Arbor, Michigan. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Avia Efrat, Uri Shaham, Dan Kilman, and Omer Levy. 2021. Cryptonite: A cryptic crossword benchmark for extreme ambiguity in language. *CoRR*, abs/2103.01242.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, William J Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Christopher Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31:59–79.

Matthew L. Ginsberg. 2014. Dr.fill: Crosswords and an implemented solver for singly weighted csps. *CoRR*, abs/1401.4597.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Michael L. Littman, Greg A. Keim, and Noam Shazeer. 2002. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134(1):23–55.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist.

Josh Rozner, Christopher Potts, and Kyle Mahowald. 2021. Decrypting cryptic crosswords: Semantically complex wordplay puzzles as a target for nlp. In *Advances in Neural Information Processing Systems*, volume 34, pages 11409–11421. Curran Associates, Inc.

Aliaksei Severyn, Massimo Nicosia, Gianni Barlacchi, and Alessandro Moschitti. 2015. Distributional neural networks for automatic resolution of crossword puzzles. In *ACL*.

A Appendix

Prompt Type (# samples)	**% Punctuation**
Zero-Shot Baseline Prompt (10K)	39.3
Zero-Shot Constrained Prompt (10K)	31.7
One-Shot Baseline Prompt (10K)	6.88
One-Shot Constrained Prompt (10K)	6.67
Three-Shot Baseline Prompt (5k)	7.12
Three-Shot Constrained Prompt (5k)	6.92

Table 3: % of outputs containing punctuation and white space across various prompt types. Few shot learning led allowed InstructGPT to learn correct crossword answer format, greatly reducing the amount of non letter characters in outputs.