Optimizing Large Language Models to Solve Crossword Puzzles

Stanford CS224N Custom Project

Ishan Mehta
Department of Computer Science
Stanford University
`ishanm@stanford.edu`

Ohm Patel
Department of Computer Science
Stanford University
`ohmpatel@stanford.edu`

Andrew Lipschultz
Department of Chemical Engineering
Stanford University
`andrewl5@stanford.edu`

Abstract

Crossword puzzles are an extremely difficult task for Large Language Models (LLMs) to complete as they require interpreting a clue (which may be fill in the blank, asking for a synonym, answering a question/metaphor, etc), utilizing a large knowledge base to answer a clue, and inputting the answers into a grid where the answers intersect correctly and, at times, follow a central theme. Thus, we sought to fine-tune various LLMs and then implement crossword solving strategies to complete the goal of solving a perfect crossword puzzle. We used the 'New York Times Crossword Clues and Answers 1993-2021' dataset from Kaggle to train and test our various models. First, we evaluated GPT-2 as a baseline as it is free, open-source, and conducive to fine tuning. When fed 781,573 crossword clues and a given character length, it never responded correctly. Next, we finetuned GPT-2 to be familiar with crossword clue/answer format, and there was an enormous improvement, with a letter accuracy of 24.42%. Realizing we needed an LLM with a larger knowledge base to make meaningful progress, we used LLAMA-2 as another baseline and it answered correctly 29.06% of the time. Finally, we fine-tuned GPT-3.5 which outputted a letter accuracy of 45.62%. We then took this fine-tuned model and utilized it to generate a list of 5 answers per clue, and used the sentence transformers library from (Reimers and Gurevych, 2019) to create clue and answer biencoders to assign probabilities to each answer candidate. With a ranked list of answers for each clue, we implemented loopy belief propagation to fill the grid, achieving a letter accuracy of 63.64%.

1 Key Information to include

TA mentor: Rashon Poole. External collaborators: No. External mentor: No. Sharing project: No.

2 Introduction

Crossword puzzles are a popular pastime for Americans; there are millions of solvers a day with the New York Times being the most popular puzzle. For nearly a century, the New York Times has published a puzzle a day with the Monday-Thursday and Sunday puzzles possessing a theme(many of the longer clues having a common thread) while the Friday/Saturday puzzles are deemed the most challenging. There are usually about 70 words per puzzle, and each grid space is shared by 2 clues. The grid is invariant under 180º rotation and is typically 15x15 although it can range up to

21 x 21. Crossword puzzles are a particularly interesting challenge for NLP as the model needs to understand what a clue is asking, retrieve the knowledge, and balance the constraints of intersecting clues. Additionally, the clues can be quite difficult, including trivia, idioms/metaphors, word play, foreign languages, visual cues, and even references to other clues in the puzzle. Early attempts to tackle this challenge used search and probabilistic methods to generate, fill, and regenerate answers. Subsequent iterations have relied on larger knowledge bases like dictionaries and Wikipedia. Most recently, LLMs have been utilized due to their enhanced understanding of language structure and access to a larger knowledge base. Thus, we fine-tuned various LLMs and implemented Loopy Belief Propagation (LBP) to fill the grid and achieve this task.

3 Related Work

3.1 Probabilistic Approach

In 2002, (Littman et al., 2002) created Proverb, a crossword solver that utilized 30 expert modules to generate a candidate list of answers and associated probabilities. Each module was specialized in searching for a specific clue type (i.e. fill in the blank vs pun) and would search the enormous database for an answer generating anywhere between 0 and 10,000 candidates. The group then created a probability distribution over the entire crossword space and maximized the product of the probabilities of each candidate to fill the grid. The model took 30 minutes and the group achieved a 98.1% letter accuracy on 70 crosswords from various sources.

3.2 Dr. Fill

In 2011, (Ginsberg, 2011) created Dr. Fill, a model that placed in the top 50 at the American Crossword Puzzle Tournament. This model solved a similar constraint satisfaction problem(with the exception of optimizing for a perfect puzzle over letter accuracy) but modified their search in many ways including: adding word order/fill-order heuristics, using limited discrepancy search, and some post-processing techniques. Furthermore, Dr. Fill utilized a far larger knowledge base including 4.8 million crossword clues (compared to 250,000 in Proverb), multiple dictionaries, and Wikipedia. They scored a 98.5% letter accuracy and 95.8% word accuracy.

3.3 Berkeley Crossword Solver

Teaming up with Mathew Ginsberg of Dr. Fill, (Wallace et al., 2022) created the Berkeley Crossword Solver, which won the American Crossword Puzzle Tournament. The team first generated a first pass list of candidate answers, only including answers the model had seen in the training data. The group then created clue and answer biencoders that are initialized as BERT-base-uncased Devlin et al. (2019) and assigned new probabilities in the encoding space, where the top-1000 recall contained 96% of the correct answers. Then, the group used loopy belief propagation to fill the grid, and used local search to make 1-2 character edits after. Finally, the clues and answers were put back into the biencoders with the new constraints to fix uncertain clues and achieved 99.9% letter accuracy on NYT puzzles with 89.5% perfect puzzles.

3.4 Tree of Thoughts

While LLMs were created for the purpose of text generation. they have proven extremely useful for logical reasoning tasks and thus could be applied to crossword puzzles. In 2023, (Yao et al., 2023) used GPT-4 to solve mini-crossword. While the researchers were more focused on implementing a Tree of Thought logical paradigm to play various games, this is the paper we first identified as using LLMs to generate answers to clues. They combined the tree of thought paradigm with a depth first search to solve mini 5x5 crosswords and achieved a 78% letter and 60% word accuracy. Our work is a promising next step as we fine-tune GPT-3.5 to excel at solving crosswords clues and utilize the clue/answer biencoders proposed by the Berkeley team. Additionally, we build upon their loopy belief propagation strategy by adding bigram probabilities. Although certain aspects of our work are building on these references, the pipeline (outlined in Figure 1) is an original pipeline, something not done end-to-end by any papers we could find.

4 Approach

4.1 Baselines

We have three baselines for solving crossword clues: GPT-2, GPT-2 fine-tuned, and Llama 2. For each model, we pass in a CSV containing crossword clues and answers, and answer length. We then asked GPT-2 base-case and Llama-2 through an API call to give the most likely answer to every clue with the prompt: "The crossword clue is (clue). The length of the answer is (length) characters. Write the answer in all caps and with no spaces". To improve upon the GPT-2 baseline, we fine-tuned the open source GPT-2 Model on 90% of the crossword clues. The clues are formatted:"(Clue), (Clue Length), (Answer)". An example clue is: "Mischief Makers, 7, RASCALS". The idea is to train the model to understand the format of the answers and the prompt as GPT-2 is only built for text generation rather than question answer. The model eventually understood the answer should be in all capitals, of the desired length, and after a integer followed by a comma. So when prompted with "Clue, Length, ". It responds with the correct formatted answer.

4.2 General Knowledge and Generating Candidates

In order to utilize a model with a larger outside knowledge base than GPT2, we called upon GPT3.5 (Turbo). We fine-tuned this model using the same data and prompt format as GPT2. We then used the GPT3.5 model to generate a list of candidates (at most 5) for every clue in a given JSON file. For each clue, we made 10 separate calls to the model and added a temperature (diversity coefficient) of 0.9 as well as a p-sampling value of 0.9. These increased the likelihood of outputting different words as potential answers and developing a more accurate and well-encompassing candidate list to pass into the biencoder.

4.3 Bi-Encoder

We utilized the Bi-Encoder model *paraphrase-MiniLM-L6-v2*, a pretrained model designed to generate embeddings to capture semantic understanding. Given a list of candidates for solving a crossword clue, our goal is to give proababilistic ratings to each candidate based on semantic similarity to the clue. In other words, rate the clues based on likelihood of being the answer. The biencoder is used to generate embeddings for the clue and the candidate list. Then, using the dot product between the embedding of the answers(E_a) and of the clues(E_c), a similarity score(SS) is computed for each candidate and the clue in the embedding space. In order to convert these scores into a probability(p_a), we take the softmax of the scores and create a mapping from each candidate to its corresponding probability. This allows us to factor in the probability of an answer in determining what letters to fill into the puzzle for that clue.

$$SS_i = E_a \cdot E_c \tag{1}$$

$$p_i = \frac{\exp(SS_i)}{\Sigma_j^n \exp(SS_j)} \tag{2}$$

4.4 Puzzle Solving Using Loopy Belief Propagation

With the candidate answers generated, we then aimed to complete the grid while balancing the fact that answers exist under a probability distribution and have to satisfy the constraints of the grid.

This input lent itself well to the LBP algorithm Murphy et al. (2013), which is used in probabilistic graphical models to compute marginal distributions at each node in a graph representing the possible states the node may take on. Nodes send messages($m_{i \to j}$) to neighbors, which are distributions representing the 26 states of characters A-Z, that represent the belief a node has about the state of the adjacent node given its own state and observed data. We define three potential functions that, when multiplied together and re-normalized, represent the marginal probability across each letter: a priori probability generated by the bi-encoder($\psi(x_i)$), the indicator function assuring across and down match($\phi(x_i, x_j)$, and bi-gram probability of adjacent letters($\xi(x_i, x_j)$. This algorithm was also used in prior work by the Berkeley Crossword Solver team (Wallace et al., 2022); however, we decided to also include the bigram probabilities. The final probability of a letter(p_{x_i}) is given by the a prior probability multiplied by all the messages it receives.

We implemented this algorithm in Loopy_BP.py[1] by constructing classes for VariableNode, FactorNode, and CrosswordSolvingGrid. A VariableNode type represents each graphical node in the crossword grid, which informally is a square block that can be filled in with some letter. Hence, a VariableNode consists of a log probability distribution across the 26 letters it can take on, and has a function that can receive a message from a neighbor node to update its own distribution. After receiving a message, the node normalizes its distribution by subtracting the log of the sum of the exponential of the probabilities across each letter (akin to performing softmax over log probabilities). A FactorNode is created for each clue in the puzzle. This class consists of a defined clue, a set of neighboring nodes, a candidate list for potential answers, and confidence ratings for the candidates. The messages received by VariableNodes, as previously mentioned, are generated and sent through FactorNodes. Based on the candidates and confidence ratings, a FactorNode sends a message to its neighboring VariableNodes. After convergence (25 iterations conducted on a single thread, which took only a couple of seconds), we greedily chose the letter for each node with the highest probability. This strategy prioritizes letter accuracy over word accuracy as their is no constraint that each letter belongs to the same candidate.

$$m_{i\to j} = \Sigma_{x_i}(\psi(x_i)\phi(x_i,x_j)\xi(x_i,x_j)\Pi_{k=\text{neighbors of i excluding j}}m_{k\to i} \tag{3}$$

$$p(x_i) = \psi(x_i) * \Pi_k m_{k\to i} \tag{4}$$

5 Experiments

5.1 Data

5.1.1 Fine-Tuning Data

The crossword data came from Kaggle[2], which came of the format (ID, clue, answer). It had 781,573 clues, which we divided into a 90/10 train/test split for fine-tuning GPT2. When fine-tuning GPT3.5, we trained on a random sample of 2000 clues and tested on a random sample of 625 clues.

5.1.2 Loopy Belief Propagation Data

In order to actually implement our LBP algorithm, we used a github dataset[3]of JSON files which included information such as clues, answers, grid representation, day of the week, and more. The information in these JSON files were essential in being able to accurately implement our solving algorithm since we could extract clues, corresponding grid positions, and answers. This JSON would be passed directly into our Crossword class, defined in CrosswordStruct.py[4]. This part of the project requires a separate dataset because spatial information regarding a 2D crossword grid was not a part of the Kaggle dataset.

Another aspect of LBP was incorporating bigram frequency. In order to construct a lookup table for bigrams, we used the English Letter Frequency Count created by Peter Norving, using the Google Corpus Data [5]

5.2 Evaluation method

To evaluate our models, we use Word Accuracy: $\frac{\text{Total Correct Answers}}{\text{Total Answers}}$, Letter Accuracy: $\frac{\text{total correct individual characters}}{\text{total number of characters}}$, Length Accuracy: $\frac{\text{Answers of Correct Length}}{\text{Total Answers}}$, and Cosine Similarity: $s = \frac{p\cdot q}{||p||||q||}$. The purpose of the first three methods are to test our crossword accuracy to see which axis we need to improve upon while the cosine similarity is used to see if our models are outputting

[1]`https://github.com/ishanmehta2/cs224N_final_project/blob/main/final_code/Loopy_BP.py`

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

[3]`https://github.com/doshea/nyt_crosswords/tree/master`

[4]`https://github.com/ishanmehta2/cs224N_final_project/blob/main/final_code/CrosswordStruct.py`

[5]`https://gist.github.com/lydell/c439049abac2c9226e53`

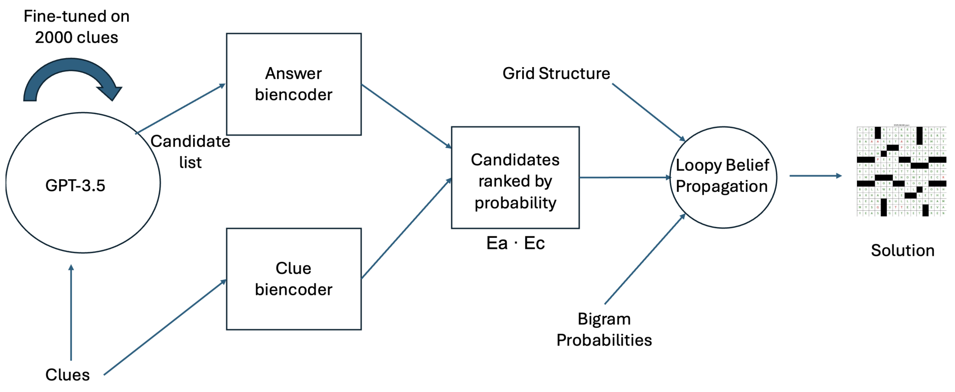


Figure 1: Crossword filling end to end

semantically similar words even when wrong. We eventually optimize for getting the highest letter accuracy when implementing loopy belief propogation.

5.3 Experimental details

5.3.1 GPT2 Fine-Tuned Baseline

We utilized a block size of 128, learning rate of 0.00001, training time of approximately 155 minutes, 3 training epochs, batch size of 3, and an AdamW optimizer.

5.3.2 GPT3.5 Fine-Tuned

We used the default parameters here which were included in API calls to openai.File.create and openai.FineTuningJob.create. We also added in a temperate of 0.9 which increased diversity of single-word outputs and a p-sample coefficient of 0.9 for further diversity. The messages were all of the format: messages=["role": "system", "content": "You are a helpful assistant.", "role": "user", "content": prompt]

5.3.3 LBP + Bi-Encoder

Our LBP model runs for 25 iterations until the whole grid is filled, using the highest probability letter for each square. After testing an assortment of iterations (10, 25, 50, 100) on a few JSON files, we saw accuracy to converge at 25 iterations. The whole end to end system, of generating candidates, building confidence ratings, and running LBP, was tested on 200 crosswords and the end to end strategy is depicted in Figure 1. The end-to-end process takes approximately 45 seconds per grid.

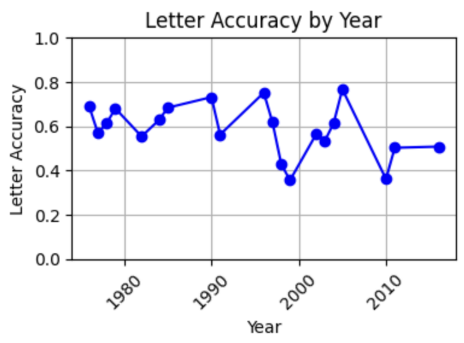
5.4 Results

	GPT2	LLAMA2	GPT2 Fine-Tuned	GPT3.5 Fine-Tuned	Bi-Encoder + Loopy BP
Word Accuracy	0.0	0.1936	0.1748	0.40	0.4327
Letter Accuracy	0.0	0.2906	0.2442	0.4562	0.6364
Length Accuracy	0.0	0.5392	0.95	0.837	N/A
Cosine Similarity (Excl Null Guesses)	N/A	0.4095	N/A	0.5138	N/A

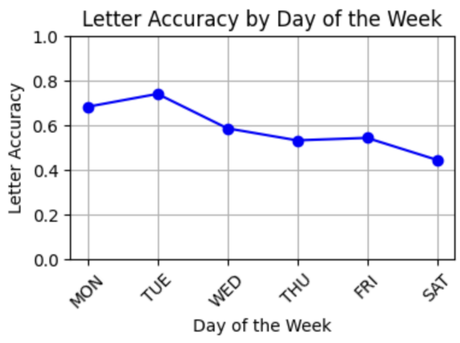
Table 1: Performance metrics for various models

The results matched our expectations of which models would perform best, but we achieved a higher accuracy than expected in the Bi-Encoder + Loopy BP. GPT-2 is not build for question/answer, so, unsurprisingly, it failed at this task. However, once we fine-tuned GPT-2 to understand the format of clues and changed to a fill in the blank format we saw substantial improvement with a word accuracy of 17.48% and a letter accuracy of 24.42%. Furthermore, it outputted a word of the correct length 95% of the time when, originally, it only outputted full length sentences with no discernible relation to the clue. Because crossword puzzles require a large knowledge base to solve, we used Llama-2 next to see if this would improve our accuracy. This model outperformed fine-tuned GPT-2 in every aspect with the exception of length accuracy. We attribute this success to the substantial general knowledge advantage Llama 2 has over a fine-tuned version of GPT-2, which helps with many trivia and factual clues. With our baselines in hand, we fine-tuned GPT-3.5 for our final attempt as it is trained on a far greater knowledge base and has an enhanced understanding of the English language. We expected this would show a great improvement over the previous models. Indeed, the word and letter accuracies for the original top answer in the candidate list were 40% and 45.2% respectively. Next, the sentence transformers re-ranked list to increase semantic association with the clue. Finally, LBP allowed us to use the grid constraints to solve harder clues given letter constrains of easier ones to achieve a final letter accuracy of 63.64% and word accuracy of 43.27%. We were surprised as this final result substantially outperformed earlier LLMs and was comparable to (Yao et al., 2023) who used GPT-4 and solved mini-puzzles as opposed to full length ones. This was the only other work we referenced that used LLMs to generate candidate answers.

We didn't include cosine similarity for the LBP algorithm because most of our mistakes were on words that didn't appear in the lexicon as the algorithm treats each cell individually. Additionally, there was no length accuracy since we padded/truncated as needed to always achieve the perfect length.

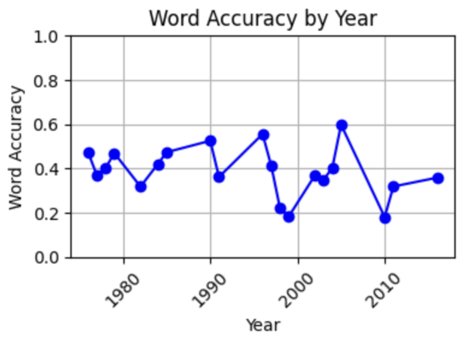


(a) Letter Accuracy by Year

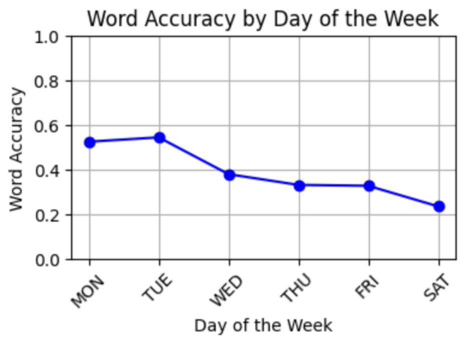


(b) Letter Accuracy by Day

Figure 2: Comparison of Letter Accuracy by Year and by Day



(a) Letter Accuracy by Year



(b) Letter Accuracy by Day

Figure 3: Comparison of Word Accuracy by Year and by Day

6 Analysis

6.1 Day of the Week

Looking at Figures 2 and 3, letter and words accuracies decrease substantially from Monday puzzles to Saturday puzzles. We attribute this to the fact that "Our difficulty scale increases through the week, with the easiest puzzles on Monday and hardest on Saturday." (New York Times, 2024)[6]. This is directly in line with our results which suggests that our fine-tuned model finds the same types of clues difficult as humans do. It can be assumed that a common knowledge model such as GPT3.5 would be better at tasks such as, fact recall or fill in the blanks, whereas puzzles later in the week include more puns, longer clues, metaphors, and other slightly more subjective and thought-specific clues.

6.2 Year

We hypothesized that the crossword solver may do better on more recent years as some of the pop culture references and slang terms the model is trained on would better reflect that of the last 10 years. However, Figures 2 and 3 demonstrate that this is not true as the model tends to do equally well across the last 40-50 years. We believe GPT-3.5 is trained on a training set that spans a larger time period than we initially thought.

6.3 Mistakes

We began analysing mistakes on our worst performing puzzle on Saturday, March 22, 2003 which had a 11.1% word accuracy. There is one specific example which highlights many of the shortcomings of our model: Clue 50 Down: 'Capital of 2.6 million'. The correct solution was 'Taipei' and our model outputted 'Naiaro'. The first issue is our candidate-generated answers are not always of the right length. Nairobi was the highest ranked answer after running through the bi-encoder. Although the population of Nairobi is not exactly 2.6 million, it is relatively close and still a capital city. The length, however, was 7 letters instead of the desired 6.

The second issue is the truncating of words. To deal with the length issue, we truncate answers which are too long and pad answers which are too short with a capital 'X' character. For this reason, 'Nairobi' turns into 'Nairob'. We decided to pad answers that were too short with 'X' because in general it wouldn't make sense at the end of the word, and thus, would penalize the words similarity score calculated with the biencoder. However, though this is true, when multiple candidates for a clue need to be padded, now the solver model with LBP has to use bigram frequencies using X, which would be very different than whatever bigrams the true letter at that place appears in.

The third issue came to light when debugging the Loopy BP algorithm. During the incorporation of bigram probabilities, we noticed at first that every cell in the grid would return a value of 'A'. The model would return 'A' when the probability was too low or there was a divide by zero error. Although we solved this error through adding a minimum threshold, there are still specific instances in which the algorithm places the letter 'A' in a cell when there is a very low probability of a specific letter value. In this case, that would come from a difficult intersection of clues across a cell. That explains the additional 'A' and eventually generated clue of 'NAIARO'. Since our candidate probabilities are not being updated after the algorithm begins running, the Loopy BP algorithm sees this answer as highly likely and alters intersecting across clues to work around the answer, leading to compounding mistakes in the grid.

Although this is just one example in one grid, it's a very telling example of some of the issues with our current implementation. Remedies could include validating length of answers before passing into LBP, re-computing probabilities after starting the algorithm, padding in a different manner, and randomizing the probability distribution key order so the letter 'A' is not slightly favored over other letters.

[6]https://www.nytimes.com/article/submit-crossword-puzzles-the-new-york-times.html?smid=url-share

6.4 Comparison With Related Works

The primary difference to be noted is between crossword solving attempts through search versus those through large language models. The first three papers listed (Proverb, Dr.Fill, and the Berkeley Crossword Solver) all utilize search mechanisms. Compared to these papers, our letter accuracy falls shy by about 25-30 points. While this seems dramatic, we also understand the difference in utilizing a search algorithm versus text generation. More comparable to our approach is the fourth paper listed (Google Deep Mind). They trump our word and letter accuracy by about 14%, but this paper used mini crosswords and GPT-4 which could also play a part in that difference. Overall, it is definitely difficult to accurately compare our results against some of the papers in the related works section due to varied methods and approaches.

7 Conclusion

7.1 Summary

Overall, we set out to utilize text generation to solve New York Times crossword puzzles. We began with three baselines in which we fed in clues and judged accuracy on similarity to the given solution: GPT2, Llama 2, and GPT2 (Fine-Tuned). With our highest letter accuracy being just under 30%, we realized the lack of general knowledge in the model was hindering our ability to achieve higher accuracy. For this reason, we fine-tuned GPT3.5 Turbo for our final, larger model. We documented the accuracies of the model itself (letter accuracy of 45.62%) before calling the model as a part of our loopy belief propagation algorithm. When combined with LBP, and a biencoder as outlined in our approaches, our letter accuracy improved to 63.64%.

7.2 Future Work

Much of our future work would revolve around improving the implementation of the algorithm to address some of the mistakes highlighted in Section 6.2. Making repeated calls to the fine-tuned model, padding length in a different manner, generating better and more accurate candidates, and assigning different probabilistic distributions to each cell could all substantially improve our accuracy. Further data such as metaphor datasets, fill in the blank datasets, and even datasets in languages such as French and Spanish would increase our model's ability to handle a larger amount of clues. We also believe that tagging clues (metaphors, puns, trivia, referential, etc) would allow us to dive deeper into exactly where our model struggles and where it excels. This type of parsing could also allow us to curate specific datasets to help achieve higher accuracies on different types of clues. Additionally, the usage of something such as Dr. Fill or another contextually-aware model could simply improve our model's accuracies on general knowledge and factual questions. With more time to parse clues/candidates/LBP, computational power, and experience, we can work to reach the accuracies of other papers and projects in this industry.

8 Ethics Statement

One of the primary ethical considerations is the fine-tuning step of our crossword solving process. We noticed on multiple occasions how closely the answers outputted after fine-tuning matched up with the training data in both format and content (see cosine similarity in results). This highlights the importance of fine-tuning such powerful and malleable models with accurate and well-formatted data. If we had fed in inaccurate facts or wrong answers, it could lead to misinformation if someone was to use our fine-tuned model. If the context of finetuning was changed to something other than a trivial word game, passing on misinformation can lead to a lot of societal damage. Already, we see impacts of humans spreading fake news and misinformation on the internet. If a LLM can be finetuned to do similar things, such an automated agent can have very harmful influences. Thus, there should be a community effort to parse wide-spread datasets to assure accuracy in fine-tuning LLMs.

Another important ethical consideration is the dominance of male voices in crossword construction. The Princeton Alumni Weekly highlights that "only 20% of Times crosswords have been created by women." (Zawistowski, 2020).[7] Our project is simply utilizing the clues as they have been written, and consequently, are susceptible to spreading and bringing more awareness to potentially gender biased clues. In future work, we could seek out publications with more diversified authors.

The final ethical consideration is the educational motivation behind crossword puzzles. Utilizing a model like ours removes the learning and improving aspects of solving crosswords manually which have been shown to increase intelligence and reduce stress. Our model is purely an experiment and we hope it isn't used in a manner which reduces learning. We could add some meta-data or a watermark to crosswords solved with our algorithm.

These issues are not specific to just crosswords, but all applications of LLMs. The primary way to mitigate the issues is based on the careful design and use of LLM systems. For example, ensuring representation and diversity in an input corpus allows for the design of a fair and ethical model. Another mitigation is through ethical use. Now, this cannot be always systematically ensured, but government regulations protecting the process of training and using language models can at least attempt to curtail harmful applications of these technologies.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Matthew L. Ginsberg. 2011. Dr.fill: Crosswords and an implemented solver for singly weighted csps. *ArXiv*, abs/1401.4597.

Michael L. Littman, Greg A. Keim, and Noam M. Shazeer. 2002. A probabilistic approach to solving crossword puzzles. *Artif. Intell.*, 134:23–55.

Kevin Murphy, Yair Weiss, and Michael I. Jordan. 2013. Loopy belief propagation for approximate inference: An empirical study.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Eric Wallace, Nicholas Tomlin, Albert Xu, Kevin Yang, Eshaan Pathak, Matthew Ginsberg, and Dan Klein. 2022. Automated crossword solving.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

9 Team Contributions

Every team member contributed equally to this project.

Ishan Mehta:

- Led GPT2 Fine-Tuning
- Led GPT3.5 Fine-Tuning with Andrew
- Data preprocessing and Github management
- Created candidate lists and implemented the biencoder for probabilities

Ohm Patel:

- CrosswordStruct design and implementation
- Designed and tested LBP algorithm with Andrew
- Llama and Gemma Baselines

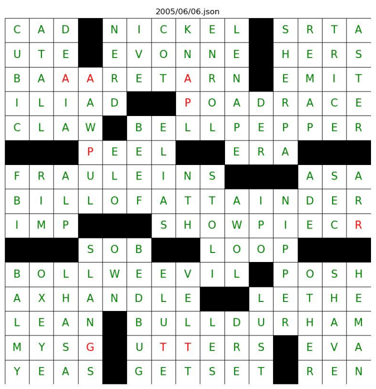
[7] https://paw.princeton.edu/article/crossword-constructors-are-mostly-male-heres-how-we-fix

- Testing Final Solver

Andrew Lipschultz:

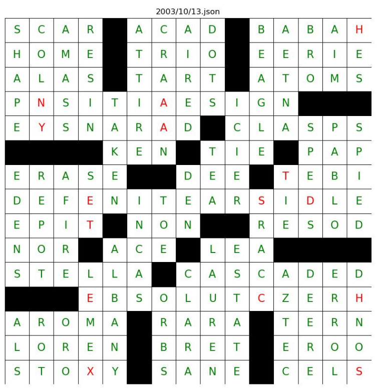
- Led GPT-2 baseline
- Deigned and tested LBP algorithm with Ohm
- Helped with GPT-3.5 fine-tuning
- Final solver solutions analysis

A Appendix (optional)


2005/06/06.json

C	A	D		N	I	C	K	E	L		S	R	T	A
U	T	E		E	V	O	N	N	E		H	E	R	S
B	A	A	A	R	E	T	A	R	N		E	M	I	T
I	L	I	A	D			P	O	A	D	R	A	C	E
C	L	A	W		B	E	L	L	P	E	P	P	E	R
			P	E	E	L			E	R	A			
F	R	A	U	L	E	I	N	S				A	S	A
B	I	L	L	O	F	A	T	T	A	I	N	D	E	R
I	M	P				S	H	O	W	P	I	E	C	R
			S	O	B			L	O	O	P			
B	O	L	L	W	E	E	V	I	L		P	O	S	H
A	X	H	A	N	D	L	E			L	E	T	H	E
L	E	A	N		B	U	L	L	D	U	R	H	A	M
M	Y	S	G		U	T	T	E	R	S		E	V	A
Y	E	A	S		G	E	T	S	E	T		R	E	N

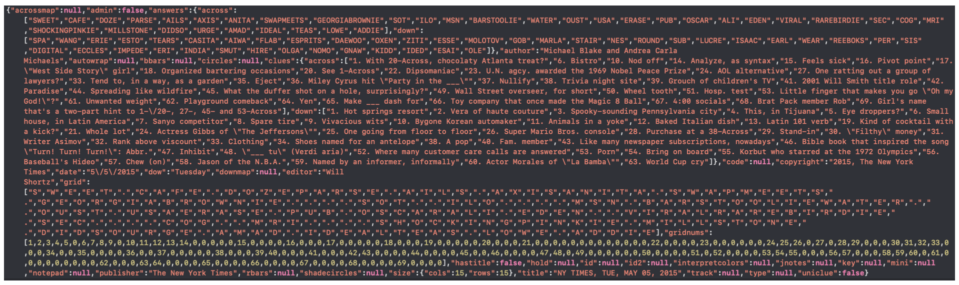
(a) June 6th, 2005 NYT Crossword


2003/10/13.json

S	C	A	R		A	C	A	D		B	A	B	A	H
H	O	M	E		T	R	I	O		E	E	R	I	E
A	L	A	S		T	A	R	T		A	T	O	M	S
P	N	S	I	T	I	A	E	S	I	G	N			
E	Y	S	N	A	R	A	D		C	L	A	S	P	S
				K	E	N		T	I	E		P	A	P
E	R	A	S	E			D	E	E		T	E	B	I
D	E	F	E	N	I	T	E	A	R	S	I	D	L	E
E	P	I	T		N	O	N			R	E	S	O	D
N	O	R		A	C	E		L	E	A				
S	T	E	L	L	A		C	A	S	C	A	D	E	D
			E	B	S	O	L	U	T	C	Z	E	R	H
A	R	O	M	A		R	A	R	A		T	E	R	N
L	O	R	E	N		B	R	E	T		E	R	O	O
S	T	O	X	Y		S	A	N	E		C	E	L	S

(b) October 13, 2003 NYT Crossword

Figure 4: Sample Completed Crosswords of High Accuracy



```
{"acrossmap":null,"admin":false,"answers":{"across":
    ["SWEET","CAFE","DOZE","PARSE","AILS","AXIS","ANITA","SWAPMEETS","GEORGIABROWNIE","SOT","ILO","MSN","BARSTOOLIE","WATER","OUST","USA","ERASE","PUB","OSCAR","ALI","EDEN","VIRAL","RAREBIRDIE","SEC","COG","MRI"
    ,"SHOCKINGPINKIE","MILLSTONE","DIDSO","URGE","AMAD","IDEAL","TEAS","LOWE","ADDIE"],"down":
    ["SPA","WANG","ERIE","ESTO","TEARS","CASITA","AIWA","FLAB","ESPRITS","DAEWOO","OXEN","ZITI","ESSE","MOLOTOV","GOB","MARLA","STAIR","NES","ROUND","SUB","LUCRE","ISAAC","EARL","WEAR","REEBOKS","PER","SIS"
    ,"DIGITAL","ECCLES","IMPEDE","ERI","INDIA","SMUT","HIRE","OLGA","NOMO","GNAW","KIDD","IDED","ESAI","OLE"]},"author":"Michael Blake and Andrea Carla
    Michaels","autowrap":null,"bbars":null,"circles":null,"clues":{"across":["1. With 20-Across, chocolaty Atlanta treat?","6. Bistro","10. Nod off","14. Analyze, as syntax","15. Feels sick","16. Pivot point","17.
    \"West Side Story\" girl","18. Organized bartering occasions","20. See 1-Across","22. Dipsomaniac","23. U.N. agcy. awarded the 1969 Nobel Peace Prize","24. AOL alternative","27. One ratting out a group of
    lawyers?","33. Tend to, in a way, as a garden","35. Eject","36. Miley Cyrus hit \"Party in the ___\"","37. Nullify","38. Trivia night site","39. Grouch of children's TV","41. 2001 Will Smith title role","42.
    Paradise","44. Spreading like wildfire","45. What the duffer shot on a hole, surprisingly?","49. Wall Street overseer, for short","50. Wheel tooth","51. Hosp. test","53. Little finger that makes you go \"Oh my
    God!\"?","61. Unwanted weight","62. Playground comeback","64. Yen","65. Make ___ dash for","66. Toy company that once made the Magic 8 Ball","67. 4:00 socials","68. Brat Pack member Rob","69. Girl's name
    that's a two-part hint to 1-\/20-, 27-, 45- and 53-Across"],"down":["1. Hot springs resort","2. Vera of haute couture","3. Spooky-sounding Pennsylvania city","4. This, in Tijuana","5. Eye droppers?","6. Small
    house, in Latin America","7. Sanyo competitor","8. Spare tire","9. Vivacious wits","10. Bygone Korean automaker","11. Animals in a yoke","12. Baked Italian dish","13. Latin 101 verb","19. Kind of cocktail with
    a kick?","21. Whole lot","24. Actress Gibbs of \"The Jeffersons\"","25. One going from floor to floor","26. Super Mario Bros. console","28. Purchase at a 38-Across","29. Stand-in","30. \"Filthy\" money","31.
    Writer Asimov","32. Rank above viscount","33. Clothing","34. Shoes named for an antelope","38. A pop","40. Fam. member","43. Like many newspaper subscriptions, nowadays","46. Bible book that inspired the song
    \"Turn! Turn! Turn!\": Abbr.","47. Inhibit","48. \"___ tu\" (Verdi aria)","52. Where many customer care calls are answered","53. Porn","54. Bring on board","55. Korbut who starred at the 1972 Olympics","56.
    Baseball's Hideo","57. Chew (on)","58. Jason of the N.B.A.","59. Named by an informer, informally","60. Actor Morales of \"La Bamba\"","63. World Cup cry"]},"code":null,"copyright":"2015, The New York
    Times","date":"5\/5\/2015","dow":"Tuesday","downmap":null,"editor":"Will
    Shortz","grid":
    ["S","W","E","E","T",".","C","A","F","E",".","D","O","Z","E","P","A","R","S","E",".","A","I","L","S",".","A","X","I","S","A","N","I","T","A",".","S","W","A","P","M","E","E","T","S","
    .","G","E","O","R","G","I","A","B","R","O","W","N","I","E",".",".",".",".","S","O","T",".",".","I","L","O",".",".",".",".","M","S","N",".","B","A","R","S","T","O","O","L","I","E","W","A","T","E","R",".","
    .","O","U","S","T",".","U","S","A","E","R","A","S","E",".","P","U","B",".","O","S","C","A","R","A","L","I",".","E","D","E","N",".",".","V","I","R","A","L","R","A","R","E","B","I","R","D","I","E","
    .","S","E","C",".",".",".",".","C","O","G",".",".","M","R","I",".",".",".",".","S","H","O","C","K","I","N","G","P","I","N","K","I","E",".","M","I","L","L","S","T","O","N","E","
    .","D","I","D","S","O","U","R","G","E",".","A","M","A","D",".","I","D","E","A","L","T","E","A","S",".","L","O","W","E",".","A","D","D","I","E"],"gridnums":
    [1,2,3,4,5,0,6,7,8,9,0,10,11,12,13,14,0,0,0,0,0,15,0,0,0,0,16,0,0,0,17,0,0,0,0,0,18,0,0,0,19,0,0,0,0,0,20,0,0,0,21,0,0,0,0,0,0,0,0,0,0,0,0,22,0,0,0,0,23,0,0,0,0,0,0,24,25,26,0,27,0,28,29,0,0,0,30,31,32,33,0
    ,0,0,34,0,0,35,0,0,0,0,36,0,0,37,0,0,0,0,0,38,0,0,0,39,40,0,0,0,41,0,0,0,42,43,0,0,0,0,44,0,0,0,0,45,0,0,46,0,0,0,0,47,48,0,49,0,0,0,0,0,0,50,0,0,0,0,51,0,52,0,0,0,0,53,54,55,0,0,0,56,57,0,0,0,58,59,60,0,61,0
    ,0,0,0,0,0,0,0,0,62,0,0,0,63,64,0,0,0,0,65,0,0,0,0,66,0,0,0,0,67,0,0,0,0,68,0,0,0,0,69,0,0,0,0],"hastitle":false,"hold":null,"id":null,"id2":null,"interpretcolors":null,"jnotes":null,"key":null,"mini":null
    ,"notepad":null,"publisher":"The New York Times","rbars":null,"shadecircles":null,"size":{"cols":15,"rows":15},"title":"NY TIMES, TUE, MAY 05, 2015","track":null,"type":null,"uniclue":false}
```

Figure 5: Example JSON