Group B12: Krishanu Datta, Emily Hahn, Xander Pero, Pranav Girish

Project Title: ClueMaster - The LLM-Powered Crossword Companion

Abstract:
Our project investigates the application of Large Language Models (LLMs) to offer solutions to New York Times crossword clues. We evaluated the effectiveness of various methods including zero-shot learning, few-shot learning, Retrieval-Augmented Generation (RAG), and fine-tuning. Through this project, we demonstrate the potential and limitations of current language technologies in addressing solving crossword clues.

ClueMaster :

The LLM-Powered Crossword Companion

Group B12- Krishanu Datta, Emily Hahn, Xander Pero, Pranav Girish

February 26, 2024

15.773 Hands-on Deep Learning

1. PROBLEM DESCRIPTION

Crossword puzzles are a daily form of entertainment that our team frequently engages with. However, we often encounter difficulties with specific clues, preventing us from completing puzzles. The common practice of revealing answers or checking puzzles for correctness is too revealing to support learning through problem-solving. In response to this issue, our team sought to develop an AI-powered crossword puzzle assistant to provide potential solutions without spoiling the answer.

Our study focused on the New York Times (NYT) crossword puzzles, noted for their distinctive clues and progressive difficulty levels. The primary aim of this project is to provide users with potential answers that are sensitive to the context and language intricacies of the clue and the structural constraints of the crossword puzzle, such as word length.

We examined various methods, including BERT, GPT, RAG and techniques for fine-tuning, and evaluated their effectiveness in navigating the linguistic terrain of NYT crossword puzzles. Through these models, we aimed to gain a deeper understanding of the capabilities of current LLM technologies and their suitability for addressing complex language-based challenges, exemplified by the nuances of NYT crossword puzzle clues.

2. APPROACH

2.1 Data Collection and Preprocessing

Our dataset consists of clues and their corresponding answers from all NYT crossword puzzles from 1993 through 2021. We augmented this data with information on word lengths, aiming to enable the models to refine their search for plausible answers. We restricted our focus to answers ranging from three to eight letters, reasoning that shorter answers are infrequent and longer ones often constitute longer phrases or rare words, which complicates prediction. We also limited our dataset to clues associated with answers that have appeared more than once, assuming that these are more likely to represent actual English words or predictable answers.

2.2 BERT

BERT.ipynb

We began our analysis by implementing a BERT model. BERT stood out to us due to its ability to pick up semantic nuances in the English language. Furthermore, its architecture allows for relatively efficient and effective fine-tuning of our dataset. As such, we posited that BERT would be relatively successful in deciphering NYT crossword clues, as solving them is contingent on the ability to decipher puns, sentiments, definitions, and other linguistic intricacies.

After fine-tuning the pre-trained model on our dataset, we ran the BERT embeddings through a neural network and determined the top 10 most likely candidate answers. The idea was that this top 10 list could be filtered by users based on known word lengths and known letters in the answer. We then recorded the accuracy, as measured by the percentage of instances in which the top 10 list contained the correct answer.

2.3 Zero-Shot Learning

Zero_Few_Shot.ipynb

After experimenting with BERT, we found that it had several limitations in its ability to interpret the nuances of NYT crossword clues effectively. As a result, we decided to pivot our focus to Large Language Models because of their ability to generate diverse and creative solutions and understand more subtleties of language.

We began with zero-shot learning, i.e., running the prompt without providing any examples or performing any fine-tuning. This would give us a general baseline understanding of the model's basic capabilities. For this task, we chose to work with the ChatGPT 3.5 Turbo API because it can interpret and respond to crossword clues directly and quickly by leveraging its extensive pre-training and diverse answer generation.

2.4 Few-Shot Learning

Building upon the zero-shot learning framework, we created a few-shot learning model that supplied additional information to enhance model performance. This method provides the ChatGPT 3.5 Turbo API with context and custom logic through examples or rules that illustrate various common crossword clue structures, such as puns, fill-in-the-blank clues, and historical references. By doing so, we enhance the model's ability to adapt its responses to the specific types of clues encountered in NYT crossword puzzles.

We tried various few-shot regimes, such as clue–answer pairs, clue–answer pairs with an explanation, and an exhaustive list of crossword clue conventions. Our goal in trying these different approaches was to see what worked and what didn't, especially in comparison to each other and zero-shot learning. We believed that each of these would improve the baseline of zero-shot learning, especially based on a given set of rules, by informing the type of answer to output.

2.4.1 Clue–Answer Pairs

The prepended phrase is:

- 'You are the best New York Times crossword solver. Clue: Fitness center? (4) Answer: CORE. Clue: M.D. org. (3) Answer: AMA. Clue: Friends of Pierre (4) Answer: AMIS. Clue: "Get out of here!" (8) Answer: LEAVENOW. Clue: "Oh, what\'s the ___?" (3) Answer: USE.'

We chose these clues because they give examples of common NYT crossword tricks, including puns, abbreviations, foreign languages, quotes, and fill-in-the-blanks. The first sentence provides context for the task at hand.

2.4.2 Clue–Answer Pairs with Explanation

The prepended phrase is:

- 'You are the best New York Times crossword solver. Clue: Fitness center? (4) Answer: The answer contains 4 letters, must be a pun because of the "?" and be a noun. Therefore, the answer is CORE. Clue: M.D. org. (3) Answer: Because of the abbreviated clue, the answer is an abbreviation. Therefore it is AMA. Clue: Friends of Pierre (4) Answer: Pierre implies a French response, and friends is a plural noun. Therefore the answer is AMIS. Clue: "Get out of here!" (8) Answer: The clue is a spoken phrase. An 8 letter phrase for this clue is LEAVENOW. Clue: "Oh, what\'s the ___?" (3) Answer: This is a fill in the blank clue. Because it is three letters and a spoken phrase, the answer is USE.'

This prompt builds upon the last in that the tricks in the clues are laid out more explicitly.

2.4.3. List of Crossword Rules

The prepended phrase is a shortened list of rules listed on Wikipedia's "*The New York Times* Crossword" page. This provides a more comprehensive list and explanation of the common language intricacies that make crossword solving such a difficult task.

2.5 Generated Knowledge Prompting

Generated knowledge prompting involves using a second model to generate knowledge about the prompt and then including that as an additional input into the primary model. In doing so, the predictive model can utilize the additional information from the initial model to make the final decision. We thought this could be useful for crossword clue generation by promoting creativity within the first section, which could be useful for clever, abbreviated, or play-on-word clues.

2.6 Retrieval-Augmented Generation (RAG)

Crosswords_RAG.ipynb

Since we had access to a large dataset of previous crossword clues and answers, it made sense to explore utilizing Retrieval-Augmented Generation (RAG) to give the LLM information about how clues have mapped to answers in the past. Because these sets were still too large for our compute memory and capacity conditions, we drew 1,000 samples for the train set and 100 for testing. We also had access to a list of styles and conventions utilized in generating New York Times crossword clues, so we used this as well.

In class, we saw how RAG can give the model access to a large corpus of data outside of its training sources to refine its responses by effectively utilizing the information provided. However, in class, we looked at more question-answer-based setups where the required answer was somewhere present in the external information source. This was not the case for our problem, and the main utility of using RAG here would be finding the clue-answer pairs that provide the maximum amount of information to the model about how it should answer the prompted clue.

For the model to access large amounts of external information without exceeding the context window, we separate the information into chunks and find the chunks that are semantically similar to the input prompt (by finding a relevance score), only providing these as part of the appended prompt. In our scenario, these chunks consisted of a clue, the number of letters in the required answer, and the corresponding answer. We explored two main methods of encoding this data.

We first encoded the dataset and the prompted clue using a pre-trained string embedding model (Sentence Transformer' all-MiniLM-L6-v2'). We then compared the encoded string using cosine similarity scores, and the most relevant clue-answer pairs were added to the prompt as examples. The key challenge with using such a model is that the data with high similarity scores are semantically similar, not logically similar. For instance, one clue for the answer “ABS” was "Fitness center?". The model focused on the semantics of the words “fitness” and “center” rather than understanding that the "?" insinuated that the clue was not literal.

To tackle this challenge, we explored an alternate setup, where the available clue-answer pairs are encoded using the styles and conventions of clues. The presence of certain critical elements of a clue (such as '?' meaning the answer is a pun or 'Abbr.' meaning that the answer is an abbreviation) are identified, and a feature vector of each clue is generated. The relevance of a past clue for a given prompted clue is now computed using a cosine similarity score on these feature vectors instead.

2.7 Fine-Tuned LLM

Finetuning LLM.ipynb

The final approach we explored to improve accuracy was fine-tuning LLMs. We used the Gemma-2B LLM model (rather than ChatGPT 3.5 Turbo) for this approach because it was more financially convenient.

In this method, we preprocessed the data similar to approaches 2.3-2.7, including removing duplicate clues and word pairings, answers longer than seven letters, and clues referencing other clues in the crossword. After applying these filters, the resulting dataset contained 525,089 data points, which yielded a training and test set of 472,580 and 52,509, respectively. Because these sets were still too large, given the compute memory and capacity conditions, 1,000 samples were drawn from this train set, while 100 samples were selected for testing.

Gemma-2B is a 2-billion-parameter base LLM model which we imported from KerasNLP. Moreover, the model has a vocab size of 256,000 tokens and uses embeddings of size 2048. To tune this base LLM on instructions, we instruction-fine-tuned the model, applying low-rank adaption (LoRA) to get better responses from the model. This tuning ultimately reduced the number of trainable parameters from 2.5 billion to 1.3 million but alleviated some of the computational load associated with fine-tuning. Finally, we fine-tuned the Gemma-2B with our given dataset of clues and answers, which acted as instructions and response pairs. More specifically, we limited the input sequence length to 512, used the AdamW optimizer, excluded layer norm and bias terms from decay, and specified one epoch and batch size of one for training.

3. RESULTS

Our exploration into leveraging different models to solve New York Times crossword puzzles yielded insightful results, particularly concerning the effectiveness of Large Language Models (LLMs) in interpreting the intricate clues that characterize these puzzles. Below is a table of the models tested and their performances measured by accuracy:

Model	**Accuracy**
BERT	0.055
Zero-Shot	0.468
Few-Shot: Clue–Answers	0.423
Few-Shot: Clue–Answers + Explanation	0.470
Few-Shot: Rules	0.452
Knowledge Generation	0.289
Knowledge Generation + Few-Shot: Clue–Answers + Explanation	0.263
RAG (with string embeddings)	0.45
RAG (with rule-based feature vectors)	0.43
Fine-Tuned	0.100

We anticipated the relatively low accuracy of BERT given its focus on semantic analysis rather than generative capabilities. This finding aligns with our initial hypothesis that while powerful for certain types of language processing tasks, BERT's architecture may not be ideally suited for the generative demands of crossword puzzle solving, where creativity and lateral thinking are often required.

Interestingly, the Zero-Shot learning approach using the ChatGPT 3.5 Turbo API performed on par with the other methods. This high performance underscores ChatGPT 3.5 Turbo's inherent capabilities due to its extensive pre-training.

We were surprised that few-shot learning performed similarly or worse than zero-shot learning. In theory, providing rules up front or a breadth of examples would brief the model, but in practice it likely distracts from the clue. Moreover, ChatGPT 3.5 Turbo had no issue with providing an answer to a clue; the issue is that it was usually wrong. Providing additional examples or rules would not alleviate this issue. For few-shot learning with clue–answer pairs and explanation, it is possible that the explanation helped the model work through the answer, matching parts of speech or clue convention similar to the provided answers. This could be similar to chain-of-thought prompting and the reason for its small success over zero-shot learning.

Generated knowledge prompting performed exceptionally poorly. We believe this is because the context from the first model could be incorrect, adding an additional layer of uncertainty and room for the thinking to go wrong. Even though the model became more creative, potentially leading it to think of multiple ways to interpret the clue, it must select a single answer before evaluation, which hinders the need for creativity.

The Retrieval Augmented Generation (RAG) showed subpar performance. Since this method utilizes a similarity score to find relevant chunks to add to the prompt, in this scenario it would look for past clues that are semantically similar, not logically similar. For this purpose, we need a more sophisticated method of computing the similarity between the prompted clue and historical clues.

The fine-tuned results are underwhelming to say the least. The fine-tuned Gemma 2D model had an out-of-sample accuracy of 10%. There are a couple of reasons which could explain this drop off from the other approaches. First and foremost, in this method, we fine-tuned Google's

Gemma 2D model rather than Open AI’s GPT 3.5 model which we used in our other methods. This difference in model choice is likely the largest factor in the approach’s lackluster performance. Additionally, with only 350 samples, 1 epoch, and a batchthe model has less data and fewer opportunities to learn from that training data. In general, this means the model may not have fully learned the underlying patterns in the data which would explain the low accuracy.

4. LESSONS LEARNED

In exploring the world of crossword puzzle solving through the lens of LLMs, our project highlights the many challenges and shortcomings of prompt engineering. Our results indicate that while LLMs hold significant promise, particularly in a Zero-Shot configuration, capturing the NYT crossword clues' creative and nuanced language remains challenging. Crafting prompts that effectively communicate the task at hand without confounding the model is complex. Despite the theoretical expectation that more advanced models or sophisticated training and fine-tuning approaches would yield progressively better results, our findings did not align with this assumption.

The results of our experiments open several paths for future improvement and experimentation. One such area involves incorporating partial answer information into the model's input. In practical settings, crossword solvers often have some letters of the answer already in place, which could significantly narrow the list of plausible solutions. This context, if effectively integrated into the model's prompt, could enhance the accuracy and relevance of the generated answers. Furthermore, the reality that a single clue may have multiple valid answers suggests that future iterations of our solution could benefit from generating a list of candidate answers rather than seeking a singular correct response.