EECS 182 Final Project SP2023

Kenneth Wang, Kevin Cai, Qingyuan Liu, Sewon Sohn

Multimodal Chain of Thought

Chain of thought (CoT) refers to the mental process of reasoning and inference that humans use to arrive at an answer or solution to a problem. It involves synthesizing information from multiple sources, making logical connections between ideas, and integrating them into a coherent line of reasoning. In the context of natural language processing and deep learning models, CoT prompting involves generating intermediate reasoning steps to arrive at the final answer to a question. This process allows the model to break down complex questions into simpler, more manageable steps, and to leverage information from multiple sources, such as text and images, to arrive at a more accurate answer. This homework assignment is heavily adapted from this paper.

This homework assignment will use the Science Question Answering (ScienceQA) dataset, comprises a total of 21,000+ multiple-choice science questions sourced from elementary and high school curricula. Through the attached notebook, you'll explore a subset of this dataset, which consists of questions that only contain an a text context as well as questions that have both text and image contexts.

This homework assignment will walk you through the sequential steps towards building a multimodal chain-of-thought model that utilizes both text and image inputs and chain-of-thought reasoning to solve ScienceQA problems.

(a) Prompt Building

Prompt building is commonly used in tasks such as text completion, translation, and question answering, where the model is required to generate output that is consistent with a given input prompt. Prompt building can be a highly effective technique for improving the accuracy and performance of language models, as it allows us to fine-tune the model's behavior for specific tasks and domains. Additionally, prompt building can help to mitigate the problem of bias in language models, as it provides a way to explicitly specify the desired output and constrain the model's behavior.

Run part (a) of the notebook to answer the following questions:

> i. This chain-of-thought model uses a two-stage framework. The first stage generates the rationale, which is trained with the "solution" text as the target. Run the block to see what the "solution" corresponds to for the question you saw above. How might this solution help the second model to get the answer?

ANSWER: The solution text should output a guide that walks through the reasoning for how one might get to the answer. For an index of 7 (the bottom feeder question), the solution text first goes describes the sturgeon and its mouth shape, then it describes looking at the two choices ("discus" and "armored catfish"). Then, the solution text describes how the armored catfish has a mouth on the underside of its head that points downwards (and is thus adapted for bottom feeding) while the discus doesn't have a mouth on the underside of its head, thus it is not adapted for bottom feeding.

As such, even without the images, the solution provides reasoning (a synonym of reasoning is rationale!) that can guide a reader to the answer. Similarly, the second model may use this reasoning (which is generated by the first model, the rationale generation model) to figure out the correct answer to "Which animal's mouth is also adapted for bottom feeding?".

ii. What are the components of the input prompt to the first stage? In other words, which pieces of a specific datapoint in the ScienceQA dataset do you concatenate together to generate the input prompt? Maintain the order that each component is attached.
hint: Look at build_train_pair.

ANSWER: Question text, newline, Context text (otherwise known as the hint in the ScienceQA dataset), newline, Choices, newline, "Answer: "

For instance, the datapoint at index 7 (the bottom feeder question) has the following components:

```
'question': "Which animal's mouth is also adapted for bottom
feeding?"
```

```
'hint': "Sturgeons eat invertebrates, plants, and small fish. They
are bottom feeders. Bottom feeders find their food at the bottom of
rivers, lakes, and the ocean.\nThe 's mouth is located on the
underside of its head and points downward. Its mouth is adapted for
bottom feeding.\nFigure: sturgeon."
```

```
'choices': ['discus', 'armored catfish']
```

Thus, the prompt built would look something like this:

```
Question: Which animal's mouth is also adapted for bottom
feeding?\nContext: Sturgeons eat invertebrates, plants, and small fish.
They are bottom feeders. Bottom feeders find their food at the bottom of
rivers, lakes, and the ocean.\nThe 's mouth is located on the underside
of its head and points downward. Its mouth is adapted for bottom
feeding.\nFigure: sturgeon.\nOptions: (A) discus (B) armored
catfish\nAnswer:
```

Note: If the problem has a corresponding image, then the context text can also append a text caption that describes the image. However, there is very minimal empirical difference, so this notebook leaves out the caption text.

(b) Add Images

Note that not all questions are associated with an image - the ScienceQA dataset comprises 10,332 (48.7%) questions with an image context, 10,220 (48.2%) with a text context, and 6,532 (30.8%) with both modalities.

i. Run part (b) of the notebook. You can try other indices and see the images as well as the questions they correspond to. To save space, this notebook only downloads certain images (question indices that produce an image are 7, 28, 45, 60). Notice that some indices produce only one image corresponding to the overall question, while other questions also produce images that correspond to the answers. Does adding the picture(s) make the question easier to solve? How might this inform our model?

ANSWER: Yes. For question 7 as an example, the question with the text itself is not intuitive, especially for people that are not familiar with fish names. However, the image added provided visual cues that were not available in the text of the question alone. It allowed for a more educated guess of the answer according to the physical appearances of the fish. Similarly, incorporating pictures into the model can improve its performance and accuracy, by allowing it to learn from both textual and visual cues. For example, a multimodal model might use a combination of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to process both the image and text data, and then use attention mechanisms to integrate the two modalities and generate an answer.

Extra: Something to note is that although multi-modality seems to help improve model performance on the ScienceQA benchmark (as well as many other similar reasoning benchmarks), it's not clear that multi-modality always improves performance, or why it seems to improve performance (beyond the intuitive idea that adding multi-modality helps human reasoning). For instance, this paper (Wu et. al. 2021, linked here) finds that some multi-modal models actually learn to *ignore* the multimodal information. They argue that multi-modality is akin to regularization, and thus can improve performance in the same way that regularization can.

Linking back to the class, you can see that multi-modality could be seen as a regularizing effect, similar to how feature augmentation to OLS is equivalent to ridge regression and thus regularizes in a similar way.

(c) Dataloading

Dataloading typically involves reading data from one or more sources, such as a file or a database, and performing preprocessing steps such as normalization, transformation, or augmentation, in order to prepare the data for use in the model.This process determines how efficiently and effectively the model can learn from the data.

i. Implement part (c) in the notebook. Initialize and tokenize the prompt and the target. Run the "Dataloader Test" code cell to check your answer.
Hint: to initialize the prompt and the target, use the given prompt building utils.

ANSWER: See notebook.

(d) Model Architecture

At a high level, the following figure shows the two stage framework of the Multimodal CoT model. The first stage involves the rationale generation, where the input is the text and image data, and the target is the "solution" (which is redefined as rationale) text you saw above. This is the first model. The second model involves answer inference, which takes as input the original text and image data with the corresponding rationale appended, and outputs the multiple choice answer (A, B, C, D, or E). Something to note is the architecture you are implementing (in the notebook, the class is called `T5ForMultimodalGeneration`) will be used twice, once for the first model (rationale generation) and once for the second model (answer inference) The following figure (Figure 1) shows the end-to-end two stage framework, which was just described in this paragraph, in pseudo-code form. Figure 2 shows a more high-level view of this two stage framework.

Input: Language input X^1_{language}, vision input X_{vision}
Output: Generated rationale R, inferred answer A
1: Construct the input $X = \{X_{\text{language}}, X_{\text{vision}}\}$
2: Generate rationale $R = F(X)$ using the model $F(\cdot)$
3: Append the rationale R to the original language input $X^2_{\text{language}} = X^1_{\text{language}} \circ R$.
4: Construct new input $X' = \{X^2_{\text{language}}, X_{\text{vision}}\}$
5: Infer the answer A by conditioning on the new input, $A = F(X')$.

Figure 1. Multimodal CoT high-level pseudocode (Zhang et al. 2023)

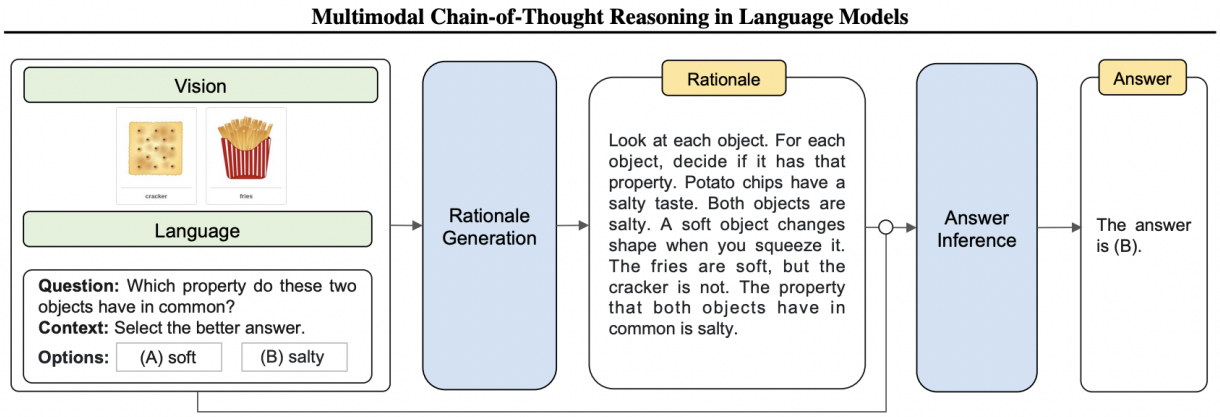

Figure 2. Multimodal CoT (Zhang et al. 2023)

This following figure (Figure 3) shows the model architecture (`T5ForMultimodalGeneration`) that you'll be implementing in part (d) of the notebook. Remember that F(X) is the model architecture (described in Figure 3 and below), which is used for both the Rationale Generation and the Answer Inference. As such, you're essentially creating two models (which have the same underlying architecture), where the first model (Rationale Generation) produces output that is appended to the original input that is then the new input that goes into the second model (Answer Inference). Below Figure 3 is a textual description of what you should implement for the code.

procedure F(X)
Encode the language and vision inputs H_{language} and H_{vision}, respectively
Build the interaction between language and vision features by attention $H^{\text{attn}}_{\text{vision}}$
Fuse H_{language} and $H^{\text{attn}}_{\text{vision}}$ by a gated fusion mechanism to have H_{fuse}
Feed H_{fuse} to the decoder to obtain the target prediction Y
return Y
end procedure

Figure 3. Pseudocode of the model architecture (Zhang et al. 2023)

Encoder:
By utilizing LanguageEncoder and VisionExtractor functions, the model can take in both language and vision inputs and generate text representation (H_{language}) and image feature (H_{vision}). The LanguageEncoder function is implemented using a Transformer model and uses the hidden states of the last layer as the language representation. We use a pre-trained model of T5Stack from HuggingFace (documentation link here) for the LanguageEncoder. On the other hand, the

VisionExtractor function vectorized the input image into vision features by extracting patch-level features. In the notebook, this is already done to save space (ie. the raw image data is not imported in, rather the ScienceQA dataset comes with the image features VisionExtractor(X_{Vision})). As such, you do not need to implement this vision extractor step.

$$H_{language} = LanguageEncoder(X_{language})$$
$$H_{vision} = W_h \cdot VisionExtractor(X_{vision})$$

Interaction:
To correlate text tokens with image patches, we use a single-head attention network after obtaining the language and vision representations. In this network, text representation is used as query (Q) and image feature is used as key (K) and value (V). After this step, gated fusion mechanism is applied, combining the text representation and image feature.

$$H_{vision}^{attn} = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$
$$\lambda = Sigmoid(W_l H_{language} + W_v H_{vision}^{attn})$$
$$H_{fuse} = (1 - \lambda) \cdot H_{language} + \lambda \cdot H_{vision}^{attn}$$

Decoder:
The resulting fused output, H_{fuse}, is then passed through the Transformer decoder to predict the target. We use the pre-trained model of T5Stack to implement the final decoder.

i. The two-stage framework takes the input (text and image), generates rationale, and then appends the rationale to the original input to create a modified input. This modified input is then passed into the second model, the inference model. What architectural structure (covered in the course) is this reminiscent of or analogous to?
Hint: If the rationale is empty or zero, what does the modified input devolve to?

ANSWER: ResNet skip connections

ii. Implement part (d) in the notebook.

ANSWER: See notebook.

(e) Visualization of rationales and training

i. What pattern do you see in the rationales generated from the model that hasn't been trained yet? Why do you think this pattern exists?

ANSWER: Some of the rationales have “zero-padding”, ie. repeated words that fill the last part of the rationale. This is because the text input is zero-padded.

Run the training cells. The T5 model is pretrained and you will finish running the last two epochs of the model. This should take about 10 minutes.

ii. With your now fully trained model, run inference on some data points! Do the rationales make sense? Do they help answer the question? Compare them to the rationales generated before the training.

ANSWER: The rationale makes sense now and they provide additional context for answering the questions. The rationales generated after training do not have the repeated words that we could see before the training.