

Building Question Decomposition Unit using Large Language Models (LLMs)

Mentor


Mihir Parmar

Parth Kachhadia

1123619202

Introduction:

Recent advances in Large Language Models have achieved state-of-the-art in many Natural Language Tasks, but there still exist certain problems for which different methods are required to achieve state-of-the-art. One such area is solving complex queries and mathematical reasoning questions. However big the Language model is, it fails to provide answers based on complex reasoning. For example, As shown in Figure 1. GPT3, which is currently the largest LM out there, fails to provide correct answers to simple maths questions which an 8-year old can answer easily. One reason for this can be attributed to how GPT3 was trained. It is a language model trained on next word prediction tasks. Hence, all it can do while generating an answer is come up with a probability distribution for words and assign the highest value as the next word generated. While other kinds of Transformer based models are trained on Language Masking tasks. Which also doesn't suit our purpose either. To overcome some of these problems, the NLP community came up with prompting methods. In which, we leverage the capacity of large models to take task specific prompts as an additional input to the model, which as name suggests allows the model to generalise on different downstream tasks. For example, GPT3 trained on LM, can be used to generate summary by providing a prompt to GPT3. Some language tasks however, would need extra layers of training, which can be solved via Fine-Tuning the pretrained LM for that particular task. For example, BERT can be fine-tuned with a classifier on Top for text-classification tasks. But, none of the above methods directly solves the problem we are trying to address. There is an approach where we change the structure of data in a way that it becomes easier for Language Model to answer questions, by decomposing complex questions into smaller units. In this project, we are using SVAMP dataset, which is elementary-level maths words problems. Objective of this project is to first develop efficient prompting techniques so that GPT3 can generate correct, high quality and useful decompositions and in second part, using selected techniques generate synthetic dataset from SVAMP, which will have decompositions. Synthetic dataset then will be used to Fine-tune BART language model, experimental results will provide understanding on whether such an approach could actually allow simple language models to answer such questions successfully.

Read the context below and provide answer to the question using mathematical reasoning. 

Context: There are 8 different books and 10 different movies in the 'crazy silly school' series. If you read 19 of the books and watched 61 of the movies.

Question: How many more movies than books are there in the 'crazy silly school' series?

Answer: 3

(Figure 1. GPT3 Fails)

Methods:

In order to understand which method of prompting would be most suitable for the problems present in SVAMP dataset. Since, problems in SVAMP dataset include mathematical reasoning questions, general idea for formulating prompts which can make GPT3 to learn to decompose questions automatically in order to arrive at true results.

Randomly selected 10 questions first decomposed manually, checking which prompting would allow GPT3 to learn to decompose questions in a meaningful way and able to answer correctly. When tried with a basic prompt with three examples of decomposition, GPT3 is able to successfully generate decompositions for new questions. But, it certainly has a problem coming up with true answers, either because of mathematical operations or the order in which these operations should be performed. This results gives an insight into which kind of prompting would allow a more efficient way. To check further, both **chain-of-thought** and **least-to-most** prompting has been tested with three examples as earlier.

Chain-of-thought prompting : In this method, we use the idea of natural language rational means a series of intermediate reasoning steps, as demonstrations using few examples to create a prompt which will allow GPT3 to learn to decompose questions.

Ex:

Context: If Lewis earns \$ 1357 every week during the 223 weeks of a harvest season.
Question: How much money will he earn if he works for 73 harvest seasons?

GPT3 Generated Output:

Ans: 73 harvest seasons * 1357 \$/week * 223 weeks/harvest season = 2,498,959.00 \$
Decomposed Question 1: How many weeks are in one harvest season?
Answer 1: 223
Decomposed Question 2: How many harvest seasons will Lewis work?
Answer 2: 73
Decomposed Question 3: How much money does Lewis earn in one week?
Answer 3: 1357
Decomposed Question 4: Multiply question_1, question_2 and question_3
Answer 4: 2,498,959.00 // wrong calculations.

Least-to-most Prompting: This prompting method works in two phases, first it divides complex questions into a set of simpler sub-questions and in phase two it solves these subproblems sequentially reaching the final answer. It has an inherent structure which suits our task of decomposing a question to answer it.

Example:

Context: If Lewis earns \$ 1357 every week during the 223 weeks of a harvest season.
Question: How much money will he earn if he works for 73 harvest seasons?

GPT3 Generated Decompositions:

Decomposed Question 1: How much money does Lewis earn every week? Decomposed Question 2: How many weeks are in a harvest season? Decomposed Question 3: How many harvest seasons will Lewis work for? Decomposed Question 4: multiplication ! "answer of subquestion 1" ! "answer of sub-question 2" ! "answer of sub-question 3"

Among the three prompting methods tried on GPT3, least-to-most and chain-of-thought both achieve significant learning capabilities with just three demonstrating examples. This augmentation method of prompting seems very fruitful especially for the task of decomposition, since the best way to learn is through examples. However, coming up with correct mathematical equations in a particular order has been challenging for GPT3 even with 3-examples provided of the tasks.

Hybrid-way: Novel approach formulated for the task, where a new instructional prompt was used to make GPT3 first learn to generate chain-of-thought reasoning to solve the question. And then use this reasoning to decompose questions correctly. Because, from above methods, chain-of-thought suited our general purpose of mathematical reasoning and provided comparative results, but it needed some instructional guidance like provided in Least-to-most technique. Given such a prompt and two examples allow GPT3 to come up with very accurate decompositions. As a last answer to the question GPT3 provides an equation, solving which is a correct answer.

Examples:

```
"context": "For the walls of the house he would use 13 large planks of wood. If each plank of wood needs 17 pieces of nails to be secured and in addition 8 nails are needed for some smaller planks."
"original_question": "How many nails does John need for the house wall?"
"reasoning": " John needs 13 large planks of wood for the walls of the house. Each large plank needs 17 pieces of nails, and in addition, he needs 8 nails for some smaller planks.",
  "decomposition": [
    {
      "question": "How many large planks of wood does John need for the house wall?",
      "answer": "13"
    },
    {
      "question": "How many pieces of nails does each large plank need?",
      "answer": "17"
    },
    {
      "question": "How many nails does John need for some smaller planks?",
      "answer": "8"
    },
    {
      "question": "How many nails does John need in total for the house wall?",
      "answer": "13*17+8"
    }
  ],
  "answer": "221.0"
```

Context: Bobby ate 38 pieces of candy. Then he ate 36 more. He also ate 16 pieces of chocolate.
Question: How many more pieces of candy than chocolate did Bobby eat?
Reasoning: Bobby ate 38 pieces of candy, then 36 more. So in total, he ate $38+36 = 74$ pieces of candy. He also ate 16 pieces of chocolate. So the difference between the amount of candy and chocolate he ate is $74-16 = 58$ pieces.

Decomposed Question 1: How many pieces of candy did Bobby eat in total?

Answer 1: 74

Decomposed Question 2: How many pieces of chocolate did Bobby eat?

Answer 2: 16

Decomposed Question 3: How many more pieces of candy than chocolate did Bobby eat?

Answer 3: $74-16$

Answer: 58.0

As part of this project, the last prompting method mentioned above has been used to generate a synthetic dataset from SVAMP dataset. GPT3 was used with a prompt to generate this dataset.

Example of dataset instance generated by GPT3.

```
"instance_43": {
  "context": "In Haley's class some boys love to play marbles. Haley gives 2 marbles to each boy. If she had 28 marbles",
  "original_question": "How many boys did she give the marbles to?",
  "reasoning": "Haley gave 2 marbles to each boy, so if she had 28 marbles, she would have given them to  $28/2 = 14$  boys.",
  "decomposition": [
    {
      "question": "How many marbles did Haley give to each boy?",
      "answer": "2"
    },
    {
      "question": "How many marbles did Haley have in total?",
      "answer": "28"
    },
    {
      "question": "How many boys would Haley have given the marbles to if she gave 2 marbles to each boy?",
      "answer": " $28/2$ "
    }
  ]
}
```

Second phase of the project involved using a pre-trained language model, fine-tuning it with a newly generated dataset for the task of question decomposition. To accomplish such a task T5 (Text-to-Text Transfer Transformer) and BART (denoising autoencoder) were considered.

Experiments:

- Selection of efficient prompting methods which can enable GPT3 to generate high quality decompositions has been performed. We already discussed all the prompting techniques tested i.e. chain-of-thought reasoning, Least-to-Most prompting, basic prompt with few-examples, Hybrid-way prompt.
- T5-small and T5-base both have been fine-tuned with synthetic data for decomposition tasks. T5 model is text-to-text transfer model which includes both encoder and decoders. But, since it is pretrained with summarization and language translation tasks, while fine-tuning, the decomposition task was modified as a summarization for T5 models. Seq2SeqTrainer function provided by hugging face has been used to fine-tune T5 model. Although training was successful, it failed to learn to decompose questions into smaller sub-question, as required by the task.

Example:

```

[18] The following columns in the training set don't have a corresponding argument in T5ForConditionalGeneration. For more details, see https://huggingface.co/docs/transformers/main_classes/sequence_to_sequence_wrapper#transformers.Seq2SeqWrapper.
/usr/local/lib/python3.7/dist-packages/transformers/optimization.py:310: FutureWarning: This implementation of T5ForConditionalGeneration is deprecated. You should use T5ForConditionalGeneration instead.
FutureWarning:
**** Running training ****
Num examples = 50
Num Epochs = 5
Instantaneous batch size per device = 16
Total train batch size (w. parallel, distributed & accumulation) = 16
Gradient Accumulation steps = 1
Total optimization steps = 20
Number of trainable parameters = 60506624
You're using a T5TokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is preferred over `encode` and `decode` methods.
[20/20 05:22, Epoch 5/5]

```

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum	Gen Len
1	No log	3.369234	25.298300	20.639000	24.008700	25.073200	16.200000
2	No log	3.181753	25.492600	20.978300	24.065100	25.263500	16.300000
3	No log	3.049766	26.265200	21.507600	24.807700	25.947800	16.650000
4	No log	2.974154	26.582600	21.786700	25.162900	26.236400	16.900000
5	No log	2.946759	26.582600	21.786700	25.162900	26.236400	16.900000

```

{'eval_loss': 1.613516092300415,
 'eval_rouge1': 29.8974,
 'eval_rouge2': 24.0332,
 'eval_rougeL': 27.754,
 'eval_rougeLsum': 29.6433,
 'eval_gen_len': 19.0,
 'eval_runtime': 15.0719,
 'eval_samples_per_second': 1.327,
 'eval_steps_per_second': 0.133,
 'epoch': 20.0}

```

context: At the zoo, a cage had snakes and alligators. The total number of animals in the cage was 79. If 24 snakes and 51 alligators were hiding.
question: How many animals were not hiding in all?

Output:

4 snakes and 51 alligator In order to find how many animals were hiding in the

- BART language model, for specific BARTForConditionalGeneration has been selected for this specific task. Because as mentioned earlier, Text Generation is facilitated by models pre-trained to summarise given text. facebook/bart-base and facebook/bart-large both have been fine-tuned using synthetic dataset generated using GPT3 to summarise given context and question into decompositions of the question with answers. In contrast to above results from T5 models, BART models learned to formulate decompositions. Although, none of the models actually generated the whole decomposition which leads to the correct answer or final mathematical equation in the last stage. The results obtained are still interesting.

Results & Analysis:

As mentioned in the project, we are using manually decomposed samples, as validation set to know, how well Pre-trained vs. Fine-tuned model performs over these examples.

BART-base:

Pre-trained evaluation on validation set:

```
{'eval_loss': 5.800280570983887,
 'eval_rouge1': 29.7113,
 'eval_rouge2': 8.1964,
 'eval_rougeL': 21.4887,
 'eval_rougeLsum': 26.2714,
 'eval_gen_len': 20.0,
 'eval_runtime': 4.1262,
 'eval_samples_per_second': 2.424,
 'eval_steps_per_second': 0.727}
```

Fine-tune Training

Step	Training Loss
50	4.030700
100	2.702200
150	2.374900
200	2.167600
250	2.011000
300	1.894200
350	1.811400

```
TrainOutput(global_step=360, training_loss=2.4096434434254963, metrics={'train_runtime': 234.5093,
 'train_samples_per_second': 5.97, 'train_steps_per_second': 1.535, 'total_flos': 426815520768000.0,
 'train_loss': 2.4096434434254963, 'epoch': 20.0})
```

After Fine-tuning on validation set:

```
{'eval_loss': 3.9140408039093018,
 'eval_rouge1': 44.7765,
 'eval_rouge2': 32.796,
 'eval_rougeL': 40.8955,
 'eval_rougeLsum': 44.975,
 'eval_gen_len': 20.0,
 'eval_runtime': 1.7839,
 'eval_samples_per_second': 5.606,
 'eval_steps_per_second': 1.682,
 'epoch': 20.0}
```

Unseen test_sample = "context: Each pack of dvds costs 107 dollars. If there is a discount of 106 dollars on each pack. original_question: How many packs of dvds can you buy with 93 dollars?"

Result:[' question: How much does each pack of dvds cost? answer: 107 question']

BART-large:

Pre-trained evaluation on validation set:

```
{'eval_loss': 6.218491554260254,
 'eval_rouge1': 28.6796,
 'eval_rouge2': 7.7421,
 'eval_rougeL': 20.5838,
```

```

'eval_rougeLsum': 25.1842,
'eval_gen_len': 20.0,
'eval_runtime': 5.965,
'eval_samples_per_second': 1.676,
'eval_steps_per_second': 0.503}

```

Fine-tune Training:

Step	Training Loss
50	4.314200
100	2.374500
150	2.038000
200	1.881200
250	1.771700
300	1.712200
350	1.680400

Training completed. Do not forget to share your model on huggingface.co/models =)

```

TrainOutput(global_step=360, training_loss=2.237399212519328, metrics={'train_runtime': 486.3313, 'train_samples_per_second': 2.879,
'train_steps_per_second': 0.74, 'total_flos': 1516973221478400.0, 'train_loss': 2.237399212519328, 'epoch': 20.0})

```

After Fine-tuning on validation set:

```

{'eval_loss': 3.828470230102539,
'eval_rouge1': 42.0875,
'eval_rouge2': 30.3097,
'eval_rougeL': 38.7522,
'eval_rougeLsum': 41.6732,
'eval_gen_len': 20.0,
'eval_runtime': 3.3412,
'eval_samples_per_second': 2.993,
'eval_steps_per_second': 0.898,
'epoch': 20.0}

```

Unseen test_sample = "context: David did 56 more push-ups than Zachary in gym class today. If David did 38 push-ups.
original_question: How many push-ups did Zachary and David do altogether?"

Result_pretrained: ['context: David did 56 more push-ups than Zachary in gym class today. original_question: Zachary did 38 push-up. If David did 38, Zachary would have done 56.original_answer: 56 push-pups.original _question: How many push-ps did Zachary and David do altogether?']

Result_fine-tuned: [' question: How many push-ups did David do in gym class? answer: 56 question : How many did Zachary? answer : 38']

Unseen test_sample = "context: Each pack of dvds costs 107 dollars. If there is a discount of 106 dollars on each pack. original_question: How many packs of dvds can you buy with 93 dollars?"

Result:[' question: How much does each pack of DVDs cost? answer: 107
question : How many packs of DVDs can you buy with 93 dollars? answer :
107-93']

As it can be identified from above samples, Both BART-base and BART-large models are able to decompose the question into a series of sub-questions. In particular, SVAMP samples contain maths word problems, and one of the efficient ways of solving such problems is to start with resolving ambiguity inherent in the question related to different values. For example, Fine-tuned models in the above case, as the text-generation process under summarisation training learns to extract different variable values as a part of decomposition. That turns out to be in-line with what GPT3 does as part of decomposition in our synthetic dataset. Although, both BART models, in the summarisation phase generate a maximum 128 token, which might be the limiting factor for these models to not generate full output required by the task.

Conclusion: This project entails a novel approach of making a simple language model learn the complex task of decomposing a mathematical reasoning question given context. Applied approach certainly proves to work on some classes of language models. Even though exact results have not been achieved, it certainly is on the right track, since partial decompositions are correct, post fine-tuning model seems to have grasp the idea of extracting values for the variables first which is needed in mathematical reasoning to reach the correct solution and in certain cases provides correct answers i.e. BART-large . If a similar approach has been used with more synthetic data at the fine-tuning stage, with more capable models and more computing resources, It will achieve expected results.

References:

- 1) Pruthvi Patel, Swaroop Mishra, Mihir Parmar and Chitta Baral. 2022. Is a Question Decomposition Unit All We Need? [arXiv:2205.1253](https://arxiv.org/abs/2205.1253).
- 2) Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark and Ashish Sabharwal. 2021. Text Modular Networks: Learning to Decompose Tasks in the Language of Existing Models. [arXiv:2009.0075](https://arxiv.org/abs/2009.0075).
- 3) Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le and Ed Chi. 2022. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. [arXiv:2250.10625](https://arxiv.org/abs/2250.10625).
- 4) Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi and Graham Neubig. 2021. Pre-train, Prompt, and Predict. [arXiv:2107.13586](https://arxiv.org/abs/2107.13586).