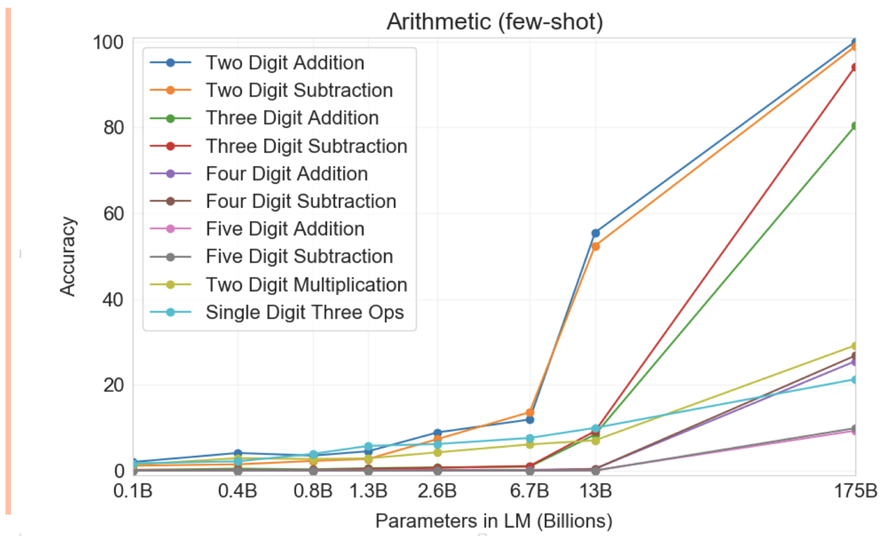
Evaluating Transformer Language Models on Arithmetic Operations Using Number Decomposition

Matteo Muffo, Aldo Cocco, Enrico Bertino @ Indigo.ai
matteo@indigo.ai, aldo@indigo.ai, e@indigo.ai

GPT-3: the first Large Language Model

- Model to generate text
- Counts 175 billions parameters
- Able to solve tasks by receiving natural language instructions
- Able to work in the zero- and few-shot settings



Research question

- GPT-3 is able to perform computations between numbers of two/three digits, but struggles when the number of digits increases
- It did not effectively learn to perform arithmetic operations
- Do Transformer Language Models have enough reasoning capabilities to learn to perform arithmetic operations of unseen numbers?

Calculon models: the idea

- Fine-tune a Transformer Language Model to perform arithmetic operations following a *decomposition pipeline*
- In this decomposition pipeline numbers are decomposed in digit form before being processed
  (e.g. 868 →8 units, 6 tens, 8 hundreds)
- This approach aims to teach calculations as children learn at school: processing units with units, tens with tens and so on by evidencing the magnitude of digits

Inference input:

"Compute with pipeline 77868 plus 4606"

Generated text at inference time:

"Translate from number to decomposition: 77868 = 8 units, 6 tens, 8 hundreds, 7 thousands, 7 tot.
Translate from number to decomposition: 4606 = 6 units, 0 tens, 6 hundreds, 4 thousands.
Sum 8 units, 6 tens, 8 hundreds, 7 thousands, 7 tot + 6 units, 0 tens, 6 hundreds, 4 thousands = 4 units, 7 tens, 4 hundreds, 2 thousands, 8 tot.
Translate from decomposition to number: 4 units, 7 tens, 4 hundreds, 2 thousands, 8 tot = 82474"

We denote as Calculon the models fine-tuned following this decomposition pipeline

Experiments

Operations tested

Additions and subtractions between numbers of 2, 3, 4 and 5 digits
Multiplications between number of 2 digits

Baselines

Baseline approach: no manipulation is performed over numbers
Example:
"Compute 7918 plus 5112 = 13030"

Spaced approach: a space is added between digits composing a number, so that each digit is tokenized singularly
Example:
"Compute 1201 plus 1302. 1 2 0 1 plus 1 3 0 2 = 2 5 0 3.
Final result = 2503"

Data

We create 9 training sets, each of which corresponding to a combination operation-approach
Addition and subtraction → 3000 observations for each number of digits tested, for a total of 12000 observations
Multiplication → 3000 observations composed by couples of 2-digit numbers exclusively

Models tested

GPT-2: start from the pre-trained gpt2-small →fine-tune it on a training set → we obtain 9 different fine-tuned models (one for each pair operation-approach)

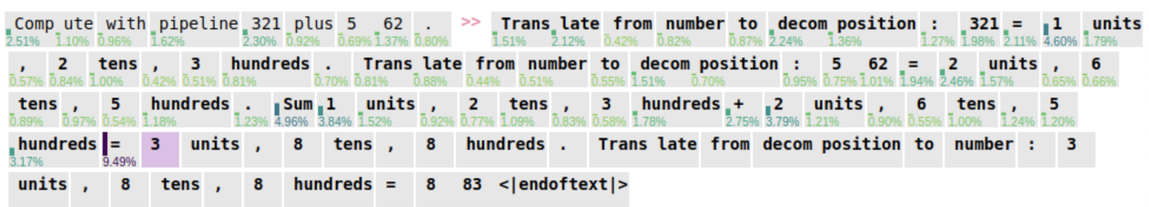
GPT-3: few-shot approach: 4 few-shot examples following the decomposition pipeline + couple of numbers to test in the input prompt, no fine-tuning

Results

Accuracy scores obtained by different models and approaches tested

Approach	2D+	3D+	4D+	5D+	2D-	3D-	4D-	5D-	2Dx
Calculon	99.75	**81.95**	**80.05**	**72.85**	**100.00**	81.35	**78.60**	**75.65**	14.65
Baseline	53.35	5.60	0.05	0.00	22.30	1.60	0.05	0.00	5.25
Spaced	90.10	77.75	67.10	57.95	45.20	11.80	1.35	0.15	5.1
GPT-3 FS	**100.0**	80.4	25.5	9.3	98.9	**94.2**	26.8	9.9	**29.2**
GPT-3 FS decomp	3.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	1.0

Input salinecy scores for a Calculon models



Conclusions

When fine-tuned with the decomposition pipeline here proposed, a Transfomer Language Model can effectively learn to perform calculations generalizing to unseen numbers

Highlighting digits magnitude can bring a remarkable benefit

GPT-3 do not show the same benefit using the decomposition pipeline in the few-shot setting

Multiplication remains an unsolved task