HW 1: CalcGPT

Submission: April 26, 2023

AIM: To disrupt the calculator industry using GPT. Instead of old-fashioned custom-built algorithms for computing answers to arithmetic problems, a system is designed and tested that uses a pre-trained LLM to solve these problems.

Method for encoding strings: The encoding of input strings is done in the encode_problems function. It takes a dataset of input values X and an encoding strategy as inputs. It outputs a list of string-encoded problems based on the selected strategy. The baseline strategy encodes the problems in the format "x1+x2=", while the experiments were done with different strategies, best performing one encodes them in the format “2+3=5 and x1+x2=”. Example- “2+3=5 and 5+6=” tokenized - {'input_ids': tensor([[17, 10, 18, 28, 20, 11, 642, 10, 21, 28]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])}.

Method for generating text: Experimented with 4 transformer models GPT 1.3B, GPT-2, GPT-2.7B, and GPT-J 6B, that are large-scale generative language model trained by EleutherAI and Huggingface. The load_LLM function loads the pre-trained models and tokenizer from the Hugging Face Transformers library. The model is set to evaluation mode and moved to the “cuda” (gpu). Temperature is set to be 0.1 and the performance declines as it’s value increases. Beams parameter was not useful as we need factually correct answer and not a more exciting or less-popular answer resulting into decreased accuracy. Min_length is set 20, max_length=50, do_sample is set to “True” and attention masks and padding is made and passed through generator according to these parameters.

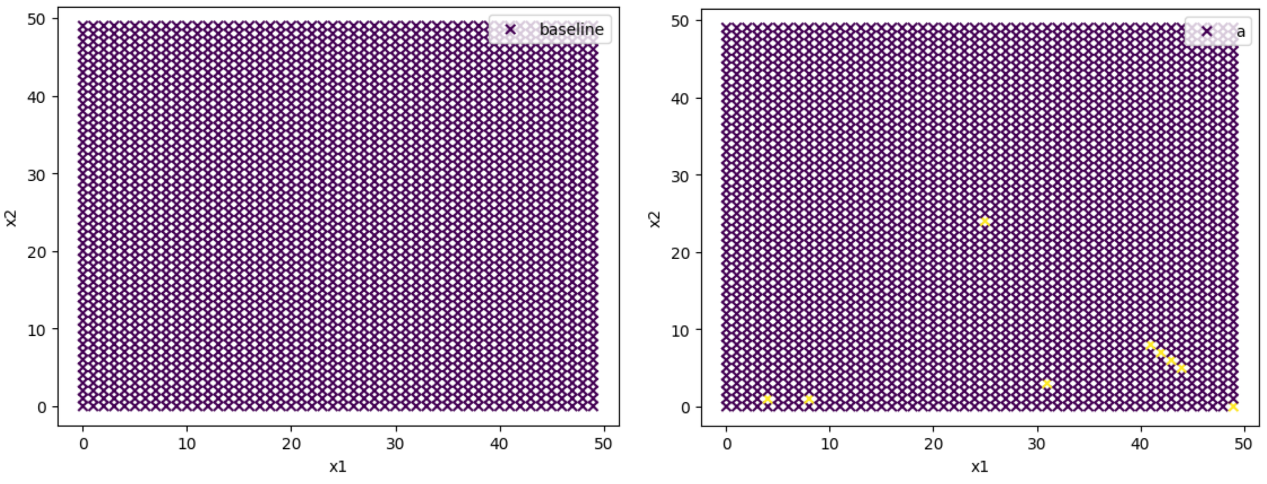
Method for decoding strings: Generated text is decoded using the tokenizer and regex to extract the integer result. For example, if the generated text is "1 plus 5 equals 6", the decoding process extracts the integer value 6. Also, torch.nan for null values is used in decoding function. The tokenized output looks like - tensor([[17, 10, 18, 28, 20, 11, 642, 10, 21, 28, 22, 13702, 198, 40, 3088, 284, 3551, 340, 355, 1061, 25, 198, 13702, 642, 10, 18, 28, 22, 11, 807, 10, 21, 28, 24, 32382, 198, 2061, 314, 1549, 588, 284, 466, 318, 910, 32382, 23, 10, 21, 28, 24]]) which is then decoded to "2+3=5, 5+6=7$$\nI tried to write it as follow:\n$$ 5+3=7, 8+6=9 $$\nWhat I'd like to do is say $$8+6=9".

Results: The dataset consists of 2500 possible operation on integers. Spaces in between math problems also played a role as without spaces, the model was performing better. Converting digits to words performed poorer with vague unrelated outputs, however good at sentence formation after the prompt. Using algebra or any kind of variables for prompts worked well only for small number upto 10. Though the comparison strategy has better accuracy of 0.2, the baseline was at 0.1. These metrics are for GPT-neo-2.7b. GPT-2.7b is converted to 8-bit quantized version producing vague errors sometimes for hugging face library functions. For GPT 1.3b the performance was really bad as accuracy was nearly 0. GPT2 is only a little better than GPT1.3b for this task as the fraction for this was also tending to zero.

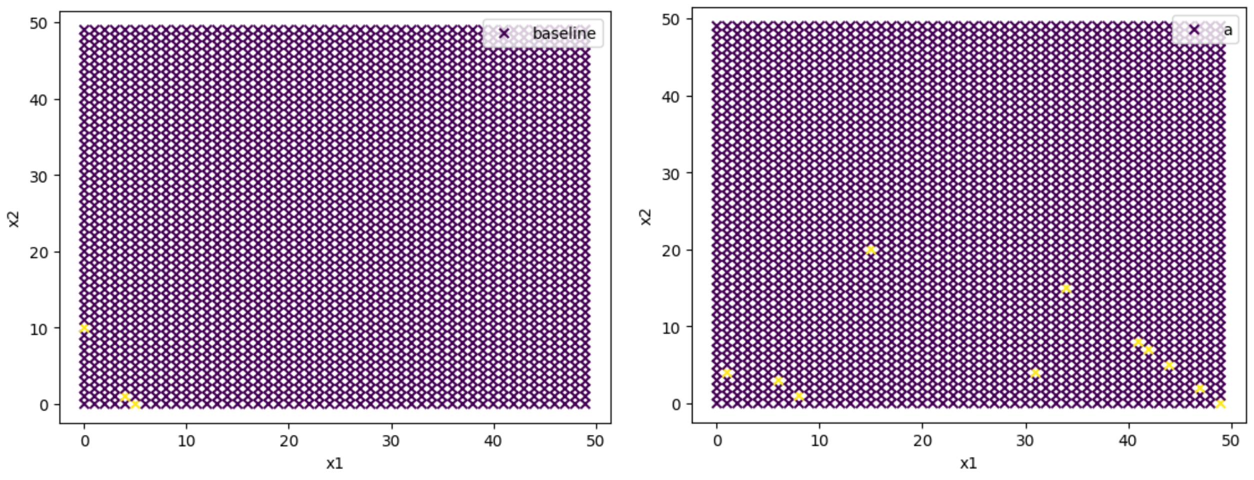
Label-

Yellow cross = correct answer & Violet cross = wrong answer

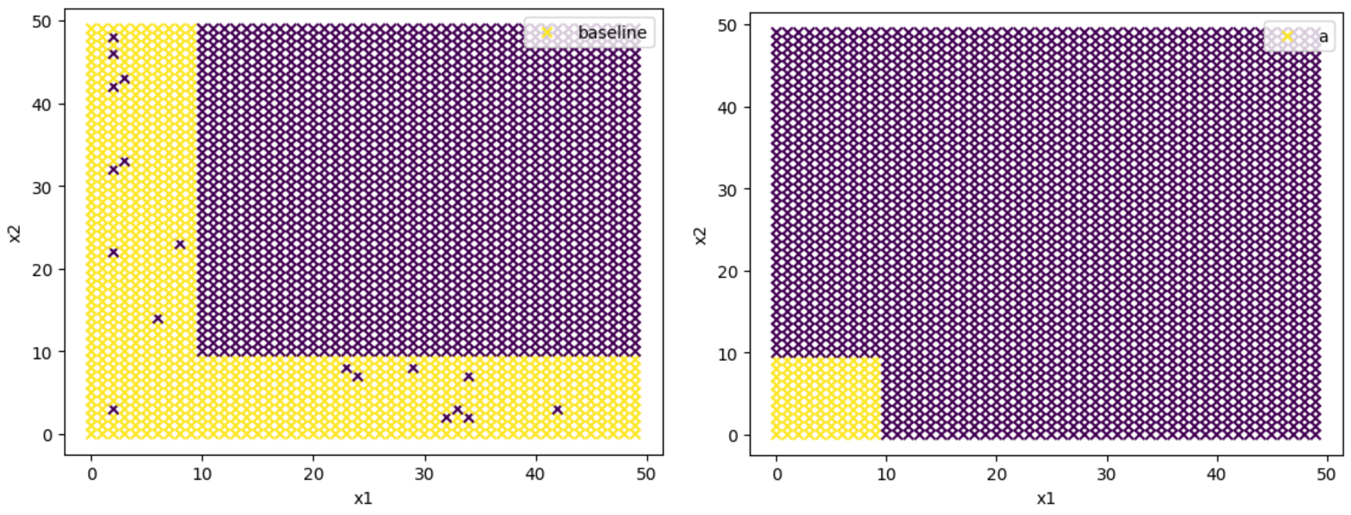
GPT neo-1.3b



GPT2 - huggingface



GPT-J 6b (8-bit quantized version)



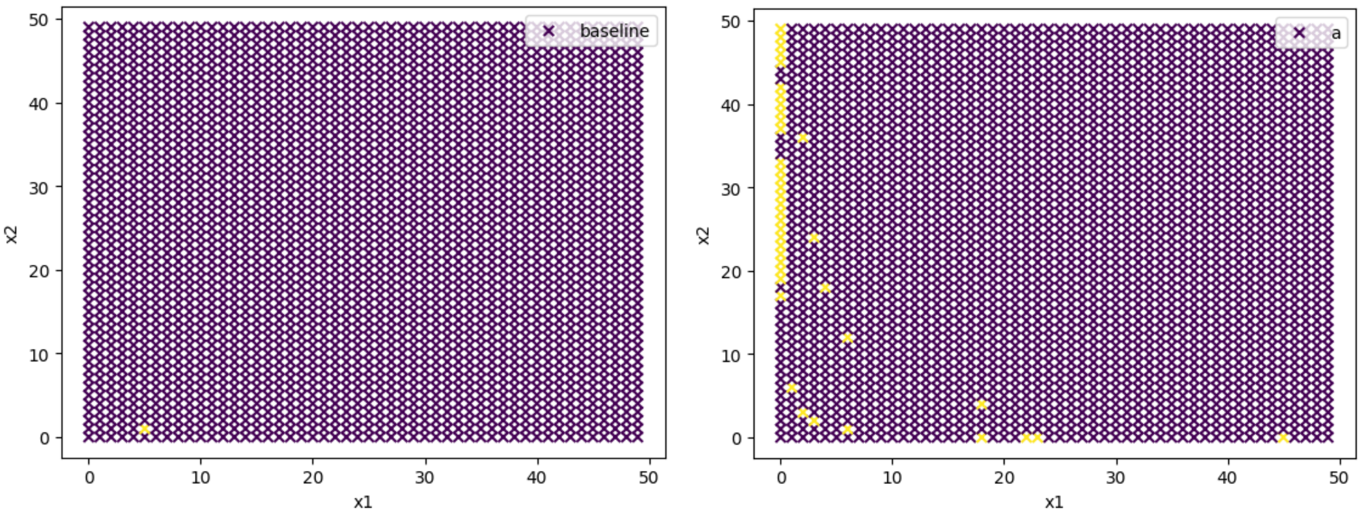
"2+3=5, 50+61=111 and 12+18=" gives no prior knowledge of 2 digit outputs for the “a” strategy resulting in a even worse accuracy of 0.04 as compared to 0.35 for baseline.

A linear SVM classifier is made using sklearn for all the models with accuracy 1, 0.988, and 0.98. Contour plots are getting complicated. However, we can easily see the pattern in inputs with respect to change in outputs. See the output of GPT-J for reference. I would have experimented on it more if google colab allowed me. I emptied all the gpu allocations for all my google accounts.
The LLMs with less parameters can be wrong randomly but if you engineer a good prompt some systematic pattern is found, which looks like an exponential function as it denotes only easier problems are correct.

AI collaboration statement: I asked chatgpt to give me the code for contour plotting, which it gave wrong and confusing. I also used it to understand some parameters of new libraries and functions.

Extra credit:
Compared other math problems (multiplication) - made the dataset and followed same instructions with GPT2. With the comparing strategy, the fraction of correct calculations turn out to be 0.02 as compared to nearly 0 for baseline. Also, the SVM linear classifier accuracy is 0.98.

GPT2 (multiplication)


baseline
a
x1
x2