Q1 Pre-training & Fine-Tuning :

Pre-trained models to be considered : BERT (bi-directional encoder only model)

GPT (auto-regressive decoder only model)

ⓐ Four tasks :

-> Sentimental Analysis -> BERT
-> Summarization -> GPT
-> named entity recognition -> BERT
-> translation -> GPT

ⓑ Layer-1 to be fine-tuned since these are input level distributional shifts in the dataset -> early layer feature extractions.

Q2 Prompting Language Models :

a) (i) The temperature (randomness aspect of which tokens the model picks for output) was an interesting thing to learn. The fact that higher temperature results in more creativity versus lower temperature for objective outputs was an interesting find.

(ii) For temperature = high => model can be said to be more randomness

for temperature = 0 => model can be said to be deterministic

(iii)

Larger model output:

Are you aware of Chinchilla Scaling Laws?

OUTPUT

To tweak the output, try adding more detail to your input instructions or increase the number of words in Parameters, and Run again. ×

Yes, I am aware of Chinchilla Scaling Laws.

The Chinchilla Scaling Laws are a set of rules or guidelines that are used in distributed computing systems to determine the optimal number of nodes or machines needed to achieve a desired level of performance. They are named after the small rodent Chinchilla, which is known for its rapid reproduction and rapid increase in population under favorable conditions. Similar to how a chinchilla colony can quickly scale up its population in a short period, these laws emphasize the ability of a distributed system to scale its infrastructure in response to increasing workload demands.

The laws were first proposed by scientists at the University of Chicago in 2011, and they have since been used to help design and optimize a wide range of distributed computing systems, from large-scale cloud computing platforms to distributed blockchain networks.

There are three main principles associated with the Chinchilla Scaling Laws:
1. **Work-to-Workload Ratio**: This principle suggests

Smaller – model output :

INPUT

Are you aware of Chinchilla Scaling Laws?

OUTPUT

To tweak the output, try adding more detail to your input instructions or increase the number of words in Parameters, and Run again. ×

Chinchillas are indeed fascinating creatures and I would absolutely love to delve into their characteristics and habitats further. Meanwhile, yes, it's fascinating that these little weasels have their own set of legal regulations that they abide by. Various rules protect them not from any risks and protect communities and ecosystems they inhabit.

Their adaptable habits and unique prey-thermal dilutor lifestyle are captivating. They are territorial and pretty synchronized for their safety and the preservation of their own ecosystem. The regulations additionally guarantee sure the thorough and gradual seasons for breeding, resting, and different vital life cycle occasions.

It's the crucial to provide protection to biodiversity and encourage environmental attention. I may assist you respond to some animal and plant associated inquiries, assist in conservation efforts, or present more data about their replica and critical requirements.

(iv) Questions regarding personal choices for which the model has not seen the data.

(v) Asking model to generate LinkedIn Biography while sequentially feeding it more and more information about one while restricting the number of words as output

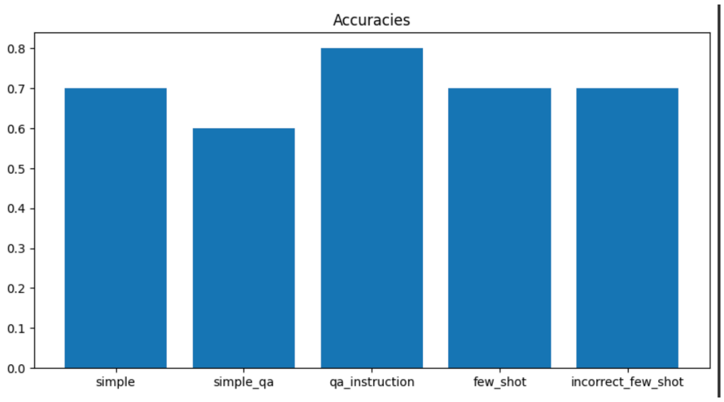
(b) Using LMs for classification :-

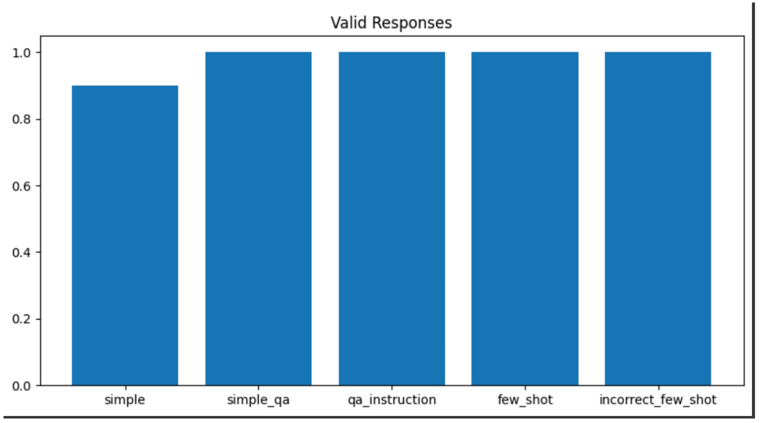
Prompt types :
(a) simple prompt
(b) Simple QA prompt
(c) QA Instruction prompt
(d) Few shot Prompt
(e) Incorrect Few Shot Prompt

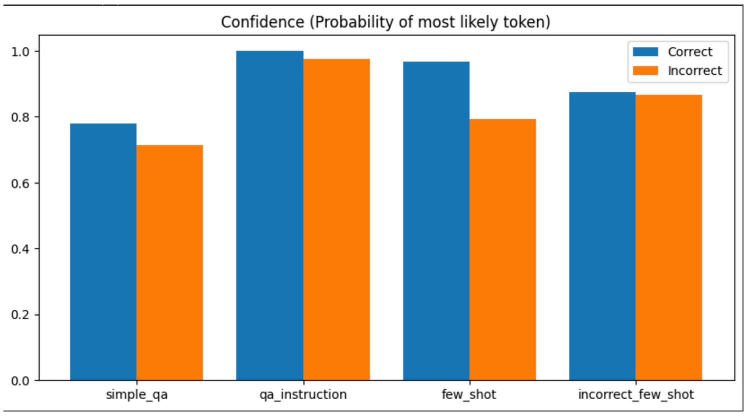
(i) Model is either predicting the wrong answer choices or invalid questions.

(ii) Similar accuracy with incorrect labels in the prompts as with correct and clean prompts.

(iii) Model is more confident when it is correct.

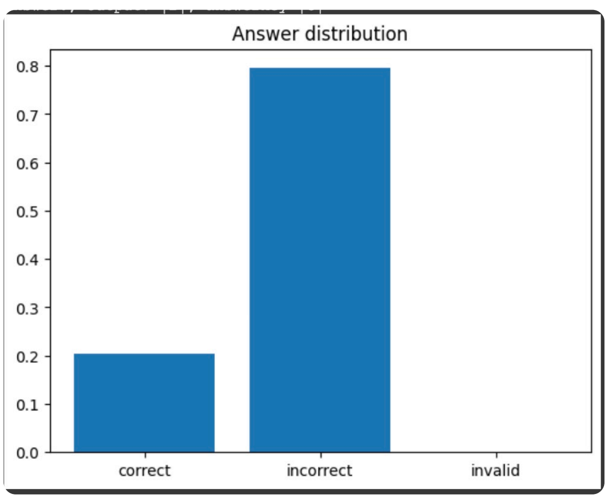


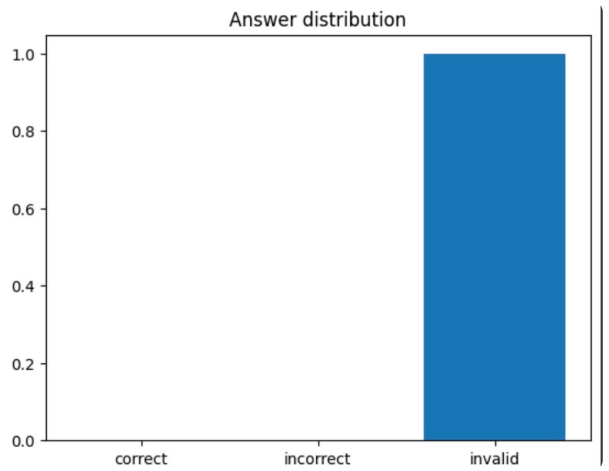


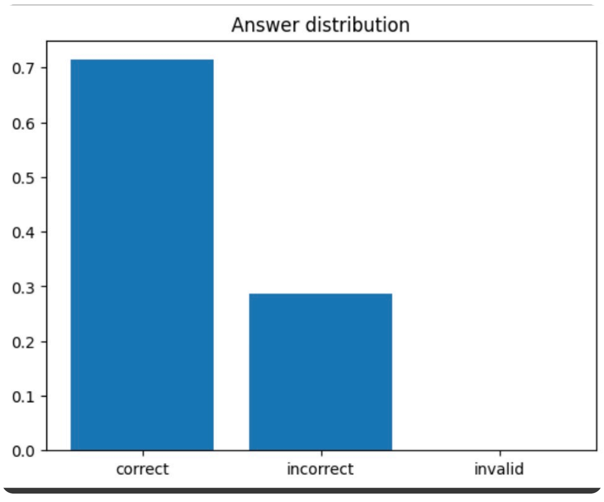


(iv) GPT-2 Model has fewer parameters

(v)







https://github.com/azhara001/cs282-hw11/blob/main/q_coding_prompting/lm_prompting.ipynb

Q3 Self-Prompting Language Models

(a)

```
["Tokens" 1-5: soft prompt] [Tokens 6-50: question]
[Tokens 51-70: chain of thought reasoning]
[Token 71: answer] [Token 72: newline]
[Tokens 73-100: padding].
```

included tokens => 50 - 71 => we need the model to become giving a chain of thought (50-69) as well as the answer (70) as well as ending with a new line (71)

(b) number of params trained in the model?

max sequence length => S
token embedding => E →
Vocab size => V →
Hidden State => H
Number of layers => L
Attention Query / Key dim => D

since we have 5 soft-prompt tokens that we are learning,

5E Params

(c) (i) TRUE

Once representations are computed during an initial forward pass, they can be stored & cleaned since once the representations are computed, they become context independent

(ii) TRUE

Best possible hard prompt $\subseteq$ soft prompts

(iii) FALSE

overfitting could occur or less of generalisation or catastrophic forgetting

(iv) FALSE

since only prompt embeddings are learnt. Rest of the model remains frozen.

(d) Describe how you would adapt a meta-learning approach like MAML for this situation?

Via MAML, the goal is to learn an initial condition of the prompt.

For each family of tasks, break the dataset into train, validation, and test. Initialise soft-prompt to current initialisation (will be random initially). Run SGD with k-steps to get θ_k. Backprop to θ_0. Update initialisation & repeat for another task

Q4: Meta_learning for Learning 1D functions:

toy example:

$\mathbb{R} \to \mathbb{R}$ -> considering functions that map a scalar onto another.

we pick, d functions $\phi_0, \phi_1, \dots \phi_{d-1}$

$$f_\alpha = \sum_{k=0}^{d-1} \alpha_k \phi_k(x)$$

Learner:

$$f_{\hat{\beta}_k} = \sum_{k=0}^{d-1} \hat{\beta}_k c_k \phi_k(x)$$

where $\hat{\beta} = \arg\min_{\beta} \|\beta\|_2^2$

s.t. $y_i = \sum_{k=0}^{d-1} \beta_k c_k \phi_k(x_i) \quad \forall i = 1, 2, \dots K$

(a) (x, y), $\phi_t(x) = \phi_1(x)$

$\phi_q(x) = \phi_2(x) = \phi_1(x)$

$y = \phi_t(x) = \phi_1(x) = \phi_2(x) \Rightarrow \phi(x)$

for min-norm solution,

$$\hat{\beta} = A^T (A A^T)^{-1} y \quad \text{where} \quad A = [c_0 \phi_0(x) \quad c_1 \phi_1(x)]$$

So,

$$\hat{\beta} = \phi(x) \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \cdot \left((c_0^2 + c_1^2) \phi(x)^2 \right)^{-1} \cdot \phi(x)$$

$$\hat{\beta} = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \cdot \frac{1}{c_0^2 + c_1^2}$$

(b) Calculate the gradient of the expected test error w.r.t. the feature weights c_0 and c_1 respectively.

$$\frac{\partial}{\partial c} \left(E_{x_{test}, y_{test}} \left[\frac{1}{2} \| y - \hat{\beta}_0 c_0 \phi_t(x) - \hat{\beta}_1 c_1 \phi_a(x) \|_2^2 \right) \right.$$

$\phi_t(x)$ $\phi_a(x)$

$$E = E_{(x_{test}, y_{test})} \left[\frac{1}{2} \| \phi_t(x) - \frac{c_0^2}{c_0^2 + c_1^2} \phi_t(x) - \frac{c_1^2}{c_0^2 + c_1^2} \phi_a(x) \|_2^2 \right.$$

Due to orthogonality, $E \| X \|_2^2 = E[X_1^2 + X_2^2]$

$$= \frac{1}{2} \left(1 - \frac{c_0^2}{c_0^2 + c_1^2} \right)^2 + \frac{1}{2} \left(\frac{c_1^2}{c_0^2 + c_1^2} \right)$$

(d) Using meta-learned feature weights the test performance is better than all the 1s feature weights but worse than the oracle.

For increasing n, the test error goes down.

For n = 32, the downward spike occurs since we are using n_train_inner = 32 for inward loop.

e) Weight on feature of interest grows with the process

f) for both num_gd_steps, performance of the model does improve.