Report : ANLP Assignment 4

2021114012
Yash Bhaskar

Theory Question

1. Concept of Soft Prompts

Addressing Limitations of Discrete Text Prompts

- **Flexibility**: Discrete text prompts are limited by a fixed set of vocabulary and rely on manual crafting, which may not capture the nuanced requirements of specific tasks. Soft prompts, being learned through backpropagation, can adapt to encapsulate subtle cues relevant to a task, beyond what predefined text prompts can offer.
- **Dynamic Adaptability**: Soft prompts can adjust their representations during training to suit the specifics of a task, something not possible with static, discrete prompts.
- **Better Task Conditioning**: Unlike discrete prompts, soft prompts aren't restricted by length or vocabulary constraints, allowing them to condition the language model more effectively for a given task.

Why Soft Prompts are More Flexible and Efficient

- **Task-Specific Tuning**: Soft prompts allow for more nuanced and specific tuning to a task, as their embeddings are optimized during training for the task at hand, rather than relying on general-purpose, pre-existing embeddings.
- **Reduced Manual Effort**: The process of finding the right discrete prompts can be labor-intensive and requires trial and error. Soft prompts eliminate this need, as they are optimized automatically.
- **Efficiency in Parameter Use**: Soft prompts can yield better performance with fewer parameters, which is a significant advantage in large-scale models where efficiency is crucial.

2. Scaling and Efficiency in Prompt Tuning

Relation Between Efficiency of Prompt Tuning and Model Scale

- **Scaling Benefits**: As language models grow in size (number of parameters), they can better leverage the nuances of soft prompts. Larger models have more capacity to integrate and utilize the fine-tuned signals from soft prompts, enhancing performance.
- **Diminishing Returns with Small Models**: Smaller models may not have sufficient capacity to fully capture the benefits of prompt tuning, leading to less pronounced improvements.

Implications for Future Developments

- **Scalability of Large Models**: The relationship suggests that as language models continue to grow, prompt tuning could become increasingly effective. This implies a promising future for developing more powerful and efficient models, especially for tasks requiring nuanced understanding or generation.
- **Adaptability to Specific Tasks**: Large-scale models equipped with prompt tuning can adapt more efficiently to specific tasks. This adaptability is critical for applications requiring bespoke responses, such as personalized content generation or complex question answering.
- **Cost-Effectiveness**: For organizations and researchers, the ability to use a single large model for multiple tasks without extensive fine-tuning (thanks to efficient prompt tuning) means more cost-effective use of resources.
- **Challenges in Resource Allocation**: While promising, this also poses challenges in computational resources, as larger models require more memory and processing power. The trade-off between model size, efficiency, and resource requirements will be a key consideration in future language model developments.

Analysis

Common Hyperparameter:

Loss Function : Cross Entropy Loss
 # Epoch : 10
Optimizer : Adam

1. Summarization Task

How Soft Prompt is Implemented

1. **Soft Prompt Creation: A custom vocabulary for the soft prompt is defined, here `[SUMMARIZE]`.**
2. **Embedding Layer: An embedding layer (`torch.nn.Embedding`) is introduced in the `GPT2WithSoftPrompt` class for the soft prompt vocabulary.**
3. **Model Extension: The class extends a pre-trained GPT-2 model and concatenates the soft prompt embeddings with GPT-2's base embeddings during the forward pass.**
4. **Training Adaptation: The model is trained to optimize only the parameters of the soft prompt embeddings, integrating the prompt into the model's responses.**

Configuration (Parameters)

- **Model: GPT-2 (`GPT2LMHeadModel`).**
- **Soft Prompt Vocabulary: `["[SUMMARIZE]"]`.**
- **Embedding Size: Default set to 768 (matching GPT-2's embedding size).**
- **Training Parameters: Batch size of 1, 10 epochs, gradient accumulation steps = 1, and gradient clipping norm = 1 are notable settings.**

Metric Used

- **Percentage of Exact Match: This metric compares the sets of output tokens and target summary tokens, calculating the percentage of output tokens that match the target tokens.**

Purpose of the Metric in Summarization

- **This metric assesses how well the model captures key information from the target summary, focusing on the accuracy of information reproduction rather than stylistic or paraphrasing elements.**

Soft Prompt

Epoch 1/10: 100%|| 2870/2870 [10:16<00:00, 3.74batch/s]
Train : % Exact Match: 16.785882918207484
Validation: 100%|| 130/130 [00:10<00:00, 10.17batch/s]
Val : % Exact Match: 17.306666893832748
Val Loss : 10.00106681964581
Epoch 2/10: 100%|| 2870/2870 [10:17<00:00, 3.72batch/s]
Train : % Exact Match: 15.878994127679539
Validation: 100%|| 130/130 [00:10<00:00, 10.17batch/s]
Val : % Exact Match: 16.60374007772575
Val Loss : 8.791603235098032
Epoch 3/10: 100%|| 2870/2870 [10:17<00:00, 3.69batch/s]
Train : % Exact Match: 15.9853698121913055
Validation: 100%|| 130/130 [00:10<00:00, 9.84batch/s]
Val : % Exact Match: 16.564957989310454
Val Loss : 8.548761661236103
Epoch 4/10: 100%|| 2870/2870 [10:18<00:00, 3.64batch/s]
Train : % Exact Match: 16.115932752210587
Validation: 100%|| 130/130 [00:10<00:00, 11.40batch/s]
Val : % Exact Match: 16.39624704438616
Val Loss : 8.436102096851055
Epoch 5/10: 100%|| 2870/2870 [10:17<00:00, 3.69batch/s]
Train : % Exact Match: 16.361446681569646
Validation: 100%|| 130/130 [00:10<00:00, 11.35batch/s]

Val : % Exact Match: 16.707541893935183
Val Loss : 8.383278700021597
Epoch 6/10: 100%|| 2870/2870 [10:22<00:00, 3.49batch/s]
Train : % Exact Match: 16.45055023666308
Validation: 100%|| 130/130 [00:10<00:00, 10.49batch/s]
Val : % Exact Match: 16.831990633783825
Val Loss : 8.323648452758789
Epoch 7/10: 100%|| 2870/2870 [10:18<00:00, 3.63batch/s]
Train : % Exact Match: 16.567110821226272
Validation: 100%|| 130/130 [00:10<00:00, 11.39batch/s]
Val : % Exact Match: 17.004581875101777
Val Loss : 8.286963316110464
Epoch 8/10: 100%|| 2870/2870 [10:20<00:00, 3.58batch/s]
Train : % Exact Match: 16.6091242059968
Validation: 100%|| 130/130 [00:10<00:00, 9.75batch/s]
Val : % Exact Match: 17.129143331716193
Val Loss : 8.246843924889198
Epoch 9/10: 100%|| 2870/2870 [10:21<00:00, 3.51batch/s]
Train : % Exact Match: 16.765729049603802
Validation: 100%|| 130/130 [00:10<00:00, 10.58batch/s]
Val : % Exact Match: 17.4535791308879523
Val Loss : 8.216329574584961
Epoch 10/10: 100%|| 2870/2870 [10:18<00:00, 3.65batch/s]
Train : % Exact Match: 17.020985130233092
Validation: 100%|| 130/130 [00:10<00:00, 9.95batch/s]
Val : % Exact Match: 17.613020407970948
Val Loss : 8.189803380232592
Test: 100%|| 110/110 [00:10<00:00, 10.04batch/s]
Test : % Exact Match: 16.58805211617323
Test Loss : 8.49093415520408

With Hard Prompt
Test: 100%|| 100/100 [00:10<00:00, 1.36s/batch]
Test : % Exact Match: 19.318181818181817
Test Loss : 11.197511672973633

The Hard Prompt Used is:
“`Summarize the following content :`”

2. Question Answer

How Soft Prompt is Implemented

5. Soft Prompt Creation: A custom vocabulary for the soft prompt is defined, here `[QUESTIONANSWERING]`.
6. Embedding Layer: An embedding layer (`torch.nn.Embedding`) is introduced in the `GPT2WithSoftPrompt` class for the soft prompt vocabulary.
7. Model Extension: The class extends a pre-trained GPT-2 model and concatenates the soft prompt embeddings with GPT-2's base embeddings during the forward pass.
8. Training Adaptation: The model is trained to optimize only the parameters of the soft prompt embeddings, integrating the prompt into the model's responses.

Configuration (Parameters)

- Model: GPT-2 (`GPT2LMHeadModel`).
- Soft Prompt Vocabulary: `["[QUESTIONANSWERING]"]`.
- Embedding Size: Default set to 768 (matching GPT-2's embedding size).
- Training Parameters: Batch size of 1, 10 epochs, gradient accumulation steps = 1, and gradient clipping norm = 1 are notable settings.

Metric Used

- Percentage of Exact Match: This metric compares the sets of output tokens and target summary tokens, calculating the percentage of output tokens that match the target tokens.

Purpose of the Metric in Question Answering

- This metric assesses how well the model captures key information from the target summary, focusing on the accuracy of information reproduction rather than stylistic or paraphrasing elements.

Soft Prompt

```
Epoch 1/10:   0%|                    | 0/5000 [10:00<?, ?batch/s]
Epoch 1/10: 100%|████████████████| 5000/5000 [11:05<00:00,  7.66batch/s]
Train : % Exact Match:  2.2524384559369075
Validation: 100%|████████████████| 5000/5000 [10:19<00:00, 25.48batch/s]
Val : % Exact Match:  2.7082617382617378
Val Loss :  11.22950127029419
Epoch 2/10: 100%|████████████████| 5000/5000 [11:00<00:00,  8.25batch/s]
Train : % Exact Match:  1.976042893008837
Validation: 100%|████████████████| 5000/5000 [10:18<00:00, 26.34batch/s]
```

Val : % Exact Match: 2.274280376486258
Val Loss : 10.746776478767394
Epoch 3/10: 100%|| 5000/5000 [10:59<00:00, 8.35batch/s]
Train : % Exact Match: 1.9270479597801582
Validation: 100%|| 5000/5000 [10:19<00:00, 26.17batch/s]
Val : % Exact Match: 2.3199665530547877
Val Loss : 10.506160795211793
Epoch 4/10: 100%|| 5000/5000 [10:59<00:00, 8.36batch/s]
Train : % Exact Match: 2.0257935740318964
Validation: 100%|| 5000/5000 [10:19<00:00, 25.85batch/s]
Val : % Exact Match: 2.3833564119507704
Val Loss : 10.309152409553528
Epoch 5/10: 100%|| 5000/5000 [10:59<00:00, 8.39batch/s]
Train : % Exact Match: 1.9569456360632829
Validation: 100%|| 5000/5000 [10:19<00:00, 25.97batch/s]
Val : % Exact Match: 2.599919100507335
Val Loss : 10.044513963699341
Epoch 6/10: 100%|| 5000/5000 [11:00<00:00, 8.30batch/s]
Train : % Exact Match: 2.162822422675364
Validation: 100%|| 5000/5000 [10:19<00:00, 25.16batch/s]
Val : % Exact Match: 2.6001054990760877
Val Loss : 9.876015647888183
Epoch 7/10: 100%|| 5000/5000 [11:01<00:00, 8.13batch/s]
Train : % Exact Match: 2.272239704739704
Validation: 100%|| 5000/5000 [10:19<00:00, 25.17batch/s]
Val : % Exact Match: 2.6334593035552096
Val Loss : 9.733137884140014
Epoch 8/10: 100%|| 5000/5000 [11:02<00:00, 7.99batch/s]
Train : % Exact Match: 2.26721250971251
Validation: 100%|| 5000/5000 [10:20<00:00, 24.96batch/s]
Val : % Exact Match: 2.548849010466656
Val Loss : 9.65255794429779
Epoch 9/10: 100%|| 5000/5000 [11:03<00:00, 7.89batch/s]
Train : % Exact Match: 2.4092756220387797
Validation: 100%|| 5000/5000 [10:19<00:00, 25.20batch/s]
Val : % Exact Match: 2.622873874491521
Val Loss : 9.587493703842163
Epoch 10/10: 100%|| 5000/5000 [11:01<00:00, 8.11batch/s]
Train : % Exact Match: 2.401964032699326
Validation: 100%|| 5000/5000 [10:19<00:00, 25.60batch/s]
Val : % Exact Match: 2.7092987567987565
Val Loss : 9.532423812866211

Hard Prompt
Validation: 100%|██████████| 500/500 [10:24<00:00, 2.06batch/s]
Val : % Exact Match: 0.8412698412698413
Val Loss : 11.775735130310059

The Hard Prompt Used is:
"`Answer the Following Question`"

3. Machine Translation

How Soft Prompt is Implemented

9. **Soft Prompt Creation: A custom vocabulary for the soft prompt is defined, here `[TRANSLATION]`.**
10. **Embedding Layer: An embedding layer (`torch.nn.Embedding`) is introduced in the `GPT2WithSoftPrompt` class for the soft prompt vocabulary.**
11. **Model Extension: The class extends a pre-trained GPT-2 model and concatenates the soft prompt embeddings with GPT-2's base embeddings during the forward pass.**
12. **Training Adaptation: The model is trained to optimize only the parameters of the soft prompt embeddings, integrating the prompt into the model's responses.**

Configuration (Parameters)

- **Model: GPT-2 (`GPT2LMHeadModel`).**
- **Soft Prompt Vocabulary: `["[TRANSLATION]"]`.**
- **Embedding Size: Default set to 768 (matching GPT-2's embedding size).**
- **Training Parameters: Batch size of 1, 10 epochs, gradient accumulation steps = 1, and gradient clipping norm = 1 are notable settings.**

Metric Used

- **BLEU Score measures the quality of the translated text by comparing it with one or more reference translations.**
- **Percentage of Exact Match: This metric compares the sets of output tokens and target summary tokens, calculating the percentage of output tokens that match the target tokens.**

Purpose of the Metric in Machine Translation

- **The BLEU (Bilingual Evaluation Understudy) Score is a widely used metric in machine translation that measures the quality of the translated text by comparing it with one or more reference translations, focusing on the accuracy and fluency of the output by assessing the presence of specific n-gram sequences.**

Soft Prompt

Epoch 1/10: 100%|| 4800/4800 [11:02<00:00, 7.69batch/s]
Train : % Exact Match: 26.7600348165731145
Train BLEU Score: 3.9490289213867651
Validation: 100%| 600/600 [10:02<00:00, 23.98batch/s]
Val : % Exact Match: 20.53414213731685
Val Loss : 13.112502169600907
Epoch 2/10: 100%| 4800/4800 [11:00<00:00, 7.94batch/s]
Train : % Exact Match: 25.6256003521302505
Train BLEU Score: 3.820488987095423e-1
Validation: 100%| 600/600 [10:02<00:00, 23.55batch/s]
Val : % Exact Match: 24.937826244198803
Val Loss : 11.866325402259827
Epoch 3/10: 100%| 4800/4800 [11:00<00:00, 7.95batch/s]
Train : % Exact Match: 26.7096354825576005
Train BLEU Score: 2.3600713497829851e-1
Validation: 100%| 600/600 [10:02<00:00, 23.63batch/s]
Val : % Exact Match: 29.25306395817232
Val Loss : 11.537898373600382
Epoch 4/10: 100%| 4800/4800 [10:59<00:00, 8.00batch/s]
Train : % Exact Match: 31.156439680269216
Train BLEU Score: 3.413226862334847e-1
Validation: 100%| 600/600 [10:02<00:00, 23.99batch/s]
Val : % Exact Match: 41.088508026201524
Val Loss : 11.779424047470092
Epoch 5/10: 100%| 4800/4800 [10:59<00:00, 8.04batch/s]
Train : % Exact Match: 33.4472728660022935
Train BLEU Score: 4.971949154134393e-1
Validation: 100%| 600/600 [10:02<00:00, 23.41batch/s]
Val : % Exact Match: 45.64973914973913
Val Loss : 11.2715537548006519
Epoch 6/10: 100%| 4800/4800 [11:00<00:00, 7.88batch/s]
Train : % Exact Match: 36.60016341582122
Train BLEU Score: 5.3278354445128505e-1
Validation: 100%| 600/600 [10:02<00:00, 23.11batch/s]
Val : % Exact Match: 50.53571428571429
Val Loss : 10.439298725128173

Epoch 7/10: 100%|| 4800/4800 [11:00<00:00, 7.91batch/s]
Train : % Exact Match: 38.49875410537181
Train BLEU Score: 4.2486008075475522e1
Validation: 100%|| 600/600 [10:02<00:00, 23.07batch/s]
Val : % Exact Match: 51.38095238095239
Val Loss : 10.010230000813802
Epoch 8/10: 100%|| 4800/4800 [10:59<00:00, 8.01batch/s]
Train : % Exact Match: 37.463936549344766
Train BLEU Score: 5.314270290674142e-1
Validation: 100%|| 600/600 [10:02<00:00, 22.70batch/s]
Val : % Exact Match: 51.583333333333336
Val Loss : 9.611779848734537
Epoch 9/10: 100%|| 4800/4800 [10:59<00:00, 8.04batch/s]
Train : % Exact Match: 39.580000915478855
Train BLEU Score: 5.333930052771299e-1
Validation: 100%|| 600/600 [10:02<00:00, 27.26batch/s]
Val : % Exact Match: 50.63888888888889
Val Loss : 9.176797207196554
Epoch 10/10: 100%|| 4800/4800 [10:56<00:00, 8.46batch/s]
Train : % Exact Match: 38.98837623218433
Train BLEU Score: 6.8040362600132474 6e-1
Validation: 100%|| 600/600 [10:02<00:00, 27.13batch/s]
Val : % Exact Match: 50.083333333333336
Val Loss : 8.861349821090698
Validation: 100%|| 600/600 [10:02<00:00, 26.09batch/s]
Test : % Exact Match: 51.24206349206349
Test Loss : 8.988617138067882

Hard Prompt

Validation: 100%|| 150/150 [10:08<00:00, 1.79batch/s]
Test : % Exact Match: 26.18131868131868
Test Loss : 11.91470266977946
Test BLEU Score: 4.298931957798081e-1

The Hard Prompt Used is:
“Translate the following sentence from english to german :”

As we can see that using Hard Prompts is not effective in accessing the actual power of generative Decoder only models. So using Prompt Tuning, we can make the model learn what Soft Prompt is best for that specific task.

Benefits

- **Flexibility and Adaptability:**
    - **Soft prompts, being learned, can adapt their embeddings to better suit the task during training, potentially leading to better performance.**
    - **Hard prompts, while easier to implement, lack this adaptability and may not capture the nuances required for complex tasks.**
- **Efficiency in Parameter Usage:**
    - **Soft prompts might use fewer parameters more efficiently, as they are optimized during training.**
    - **Hard prompts are limited by the fixed vocabulary and structure, which might not be optimal for the task.**
- **Manual Effort and Scalability:**
    - **Designing effective hard prompts can be labor-intensive and less scalable.**
    - **Soft prompts automate the optimization process, making them more scalable for multiple tasks.**
- **Interpretability and Control:**
    - **Hard prompts offer more interpretability and control, as they are manually designed.**
    - **Soft prompts are less interpretable due to their learned nature.**

Conclusion

The comparison between soft and hard prompts highlights the trade-offs between adaptability and control. While soft prompts offer dynamic adaptability and efficient parameter usage, hard prompts provide more direct interpretability and control over the model's conditioning. The choice between soft and hard prompts may depend on the specific requirements of the task, the desired level of control, and the availability of resources for manual prompt design.