ANLP Assignment 3 Report

Name: Aman Raj

Roll No: 2021101064

Theory Questions

1. **How does the introduction of "soft prompts" address the limitation of discrete text prompts in large language models? Why might soft prompts be considered as a more flexible approach for task specific conditioning?**

Ans.

Soft prompts are a way to guide language models using learned settings instead of fixed text prompts. Unlike regular text prompts, which use specific words, soft prompts use adjustable embeddings (or settings) that can better fit a task. This makes them more flexible and efficient because only the prompt needs adjusting, not the whole model. Soft prompts allow for more precise control and can capture complex instructions, making them especially useful for specific tasks.

2. **How does the efficiency of prompt tuning relate to the scale of the language model? Discuss the implications of this relationship for future developments in large-scale language models and their adaptability to specific tasks.**

Ans.

The efficiency of prompt tuning becomes more advantageous as the language model scales up. In larger models, tuning all parameters for a specific task is resource-intensive and time-consuming, often requiring significant memory and computational power. Prompt tuning, however, only adjusts a small set of learnable prompt embeddings rather than the full set of model parameters, making it much more efficient. This allows even massive models to be adapted to specific tasks without the cost of full fine-tuning.

As language models continue to grow, this efficiency opens doors for more accessible and adaptable models. It means that even large-scale models can be quickly tailored to a variety of tasks without requiring extensive resources, supporting faster innovation and broader application. Additionally, this approach could pave the way for versatile "foundation" models that can be flexibly tuned for specialized domains, making AI tools more customizable and usable across fields.

3. **What are the key principles behind Low-Rank Adaptation (LoRA) in fine-tuning large language models? How does LoRA improve upon traditional fine-tuning methods regarding efficiency and performance?**

Ans.

Low-Rank Adaptation (LoRA) fine-tunes large language models by adjusting only a small, low-rank subset of parameters rather than the entire model. It does this by adding lightweight matrices to capture essential task-specific changes, which keeps memory and computational needs low.

LoRA is more efficient than traditional fine-tuning because it uses fewer resources while still allowing the model to adapt well to new tasks. This makes it a practical way to customize large models quickly and effectively, even on limited hardware.

4. **Discuss the theoretical implications of introducing low-rank adaptations to the parameter space of large language models. How does this affect the expressiveness and generalization capabilities of the model compared to standard fine tuning?**

Ans.

Low-rank adaptations (LoRA) in large language models focus parameter updates on essential patterns, using fewer resources while still capturing key task-specific adjustments.

This approach slightly limits expressiveness compared to full fine-tuning but often improves generalization by preventing overfitting. LoRA effectively balances model specificity and adaptability, making it efficient for tuning large models across diverse tasks.

Implementation & Training

Each of these models was trained for 5 epochs using Adam optimizer with learning rate 5×10^{-4}.

The batch size for the training was set to 10, to avoid causing memory issues on Kaggle. The maximum length was set to 256 as well, in the GPT-2 tokenizer.

The models were trained on entire dataset.

Results of Training the Models:

1. LoRA model

Hyperparameters chosen: $\alpha = 16$ (scaling factor), $\mathrm{r} = 8$ (rank of matrix)

		Before Training	After Training
Training Set	Avg. Loss	12.21	3.93
	ROUGE-1	29.65	42.72
	ROUGE-2	11.69	19.37
	ROUGE-L	19.30	24.56
Evaluation Set	Avg. Loss	12.46	4.31
	ROUGE-1	30.12	45.73
	ROUGE-2	12.23	22.41
	ROUGE-L	20.91	27.76

2. Traditional Fine Tuning

		Before Training	After Training
Training Set	Avg. Loss	12.21	4.68
	ROUGE-1	29.65	40.68
	ROUGE-2	11.69	18.14
	ROUGE-L	19.30	25.71
Evaluation Set	Avg. Loss	12.46	5.34
	ROUGE-1	30.12	42.33
	ROUGE-2	12.23	19.95
	ROUGE-L	20.91	27.25

3. Soft Prompt Tuning

Hyperparameters chosen: Prompt length = 10, random initialization

		Before Training	After Training
Training Set	Avg. Loss	12.21	2.87
	ROUGE-1	29.65	12.77
	ROUGE-2	11.69	5.23
	ROUGE-L	19.30	8.35
Evaluation Set	Avg. Loss	12.46	3.21
	ROUGE-1	30.12	13.58
	ROUGE-2	12.23	6.21
	ROUGE-L	20.91	10.49

Resource Utilization for each of the models:

	No of parameters		Time taken for training
	Total	Trainable	(per epoch)
Traditional Fine Tuning	124439808	38597376	26 mins 7 secs
LoRA	124882176	442368	27 mins 32 secs
Soft Prompt	124447488	7680	25 mins 56 secs

Analysis:

- **LoRA:** LoRA demonstrated the best performance in terms of ROUGE scores, particularly on the evaluation set, suggesting superior generalization and adaptation to the task. It outperformed the Traditional Fine tuning by a good margin.
- **Traditional Fine tuning:** The traditional fine tuning is competitive and works. It had lower training time than LoRA but lower performance than LoRA as well.
- **Soft Prompt:** It performed significantly bad compared to LoRA and Traditional but if trained for more epochs, it could possibly attain a competitive level as them. It had extremely few parameters, but this compromised the adaptation very much.

Best Performance: LoRA