Soft Prompts

Estimated time needed: 25 minutes

Learning Objective:

- Describe the concepts of soft prompts for modifying input data to generate desired outputs
- Identify the best practices and benefits of using soft prompts

Introduction

In AI language models, prompting is a critical technique used to guide models in generating specific outputs. As language models grow larger and more complex, efficient training methods become crucial. One such method is prompting, which primes a frozen pre-trained model for specific tasks. Traditionally, people use "hard prompts," consisting of explicit textual instructions given to the model. However, "soft prompts" represent a more nuanced approach, offering flexibility and potential improvements in model performance. Integrating soft prompts helps fine-tune the models for specific tasks or domains, making them versatile tools in the toolkit of any natural language processing (NLP) engineer.
This reading explores the concept of soft prompts, differentiates them from hard prompts, and delves into three prominent methods: Prompt-tuning, P-tuning, and Prefix tuning.

Hard prompting vs. soft prompting

The key difference lies in the learning process. During training, the system optimizes soft prompts to improve model performance on specific tasks, while people manually craft hard prompts.

Hard prompting

Hard prompts involve manually crafted text instructions that guide the model. They are composed of discrete input tokens, which can be effective but require significant effort to design well.

Example:
To get a model to summarize text, you might use a hard prompt like:
“Summarize the following text: [Text]”

Soft prompting

Soft prompts, in contrast, use learnable tensors concatenated with input embeddings. These "virtual tokens" are optimized to the data set, providing more efficient and flexible task adaptation.

Example:
Instead of explicit text ("Summarize the following text:", in this case), a soft prompt involves adjusting input embeddings that subtly guide the model toward generating summaries.

Implementing soft prompts

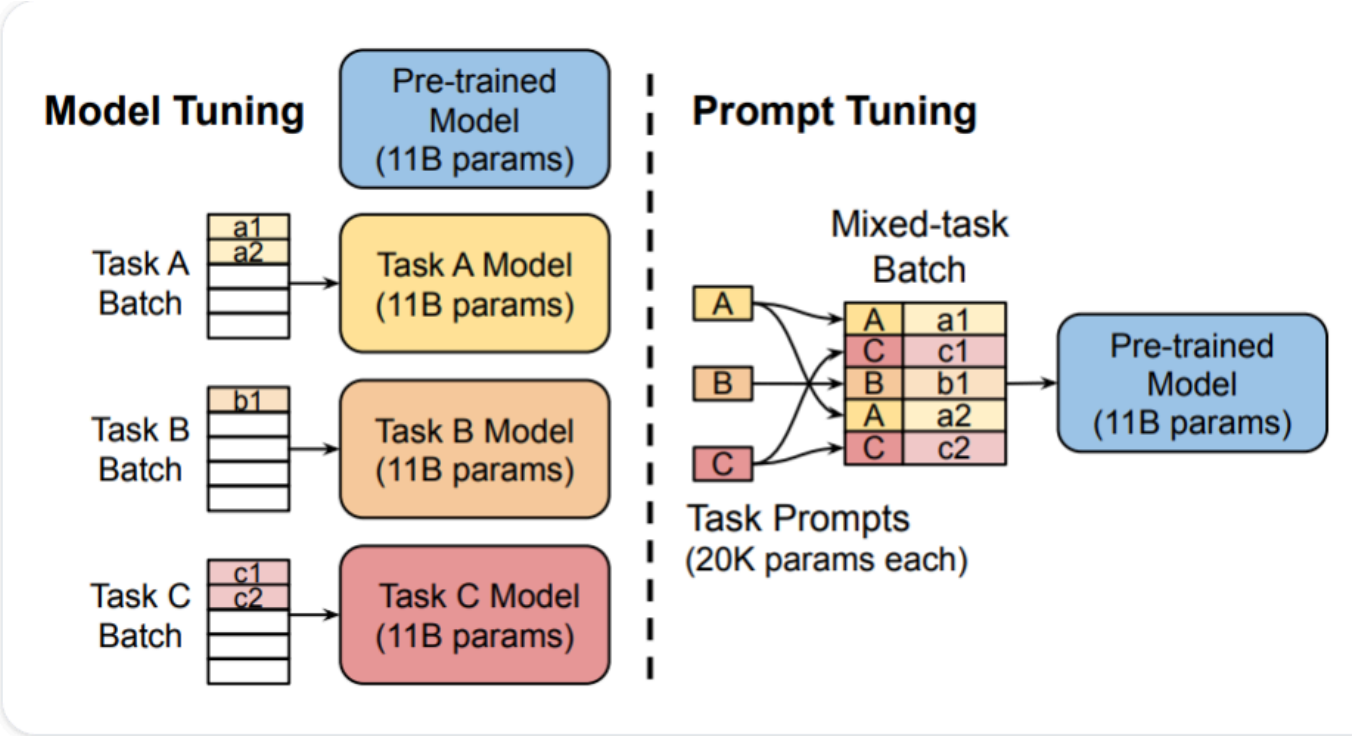
Let's explore the steps involved in implementing soft prompts:

1. **Select a pre-trained model:** Choose a large language model that will serve as the base. This model remains mostly unchanged during the soft-prompting process.
2. **Define the task:** Clearly specify the downstream task you want to perform, such as sentiment analysis, summarization, or translation.
3. **Initialize soft prompts:** Create learnable tensors (soft prompts) that will be concatenated with the input embeddings. These tensors are initialized randomly or based on prior knowledge.
4. **Integrate soft prompts:** Modify the input processing pipeline to include the soft prompts. For methods like prefix tuning, prepend these prompts to the input data.
5. **Freeze model parameters:** Keep the main model parameters frozen. Only the soft prompts (learnable tensors) are updated during training.
6. **Optimize soft prompts:** Train the model using a data set relevant to the task. During this process, optimize the soft prompts using backpropagation to improve task performance.
7. **Evaluate and tune**
   • **Evaluate:** Test the modified model on validation data to assess performance.
   • **Tune:** Adjust the learning rate or other hyperparameters as necessary to enhance results.

Methods of soft prompting

1. Prompt-tuning

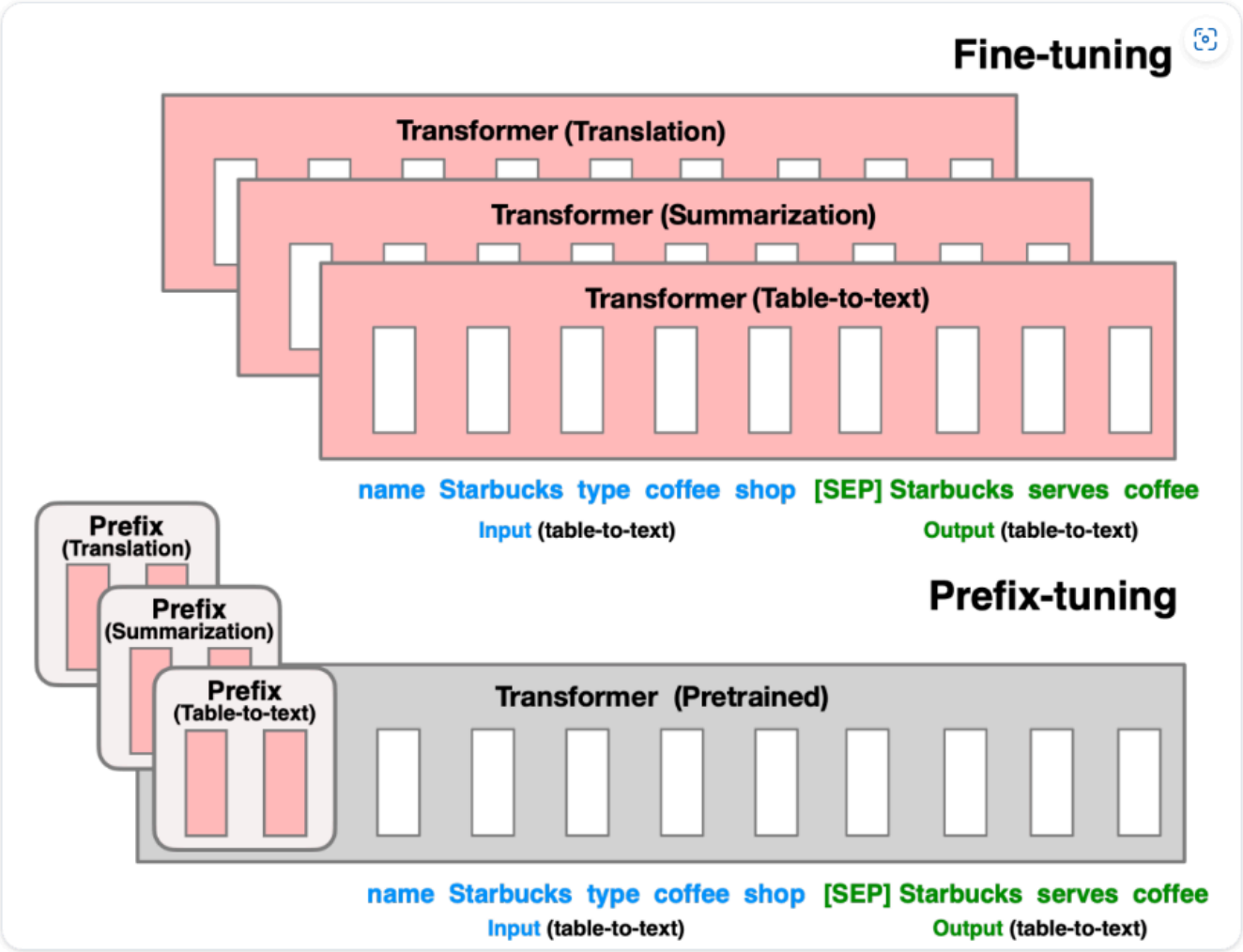
Prompt-tuning involves adding learnable parameters (soft prompts) to the input embeddings while keeping the rest of the model's parameters fixed. This method allows efficient adaptation to specific tasks without retraining the entire model. The system will freeze the pre-trained model's parameters and update only the prompt token embedding gradients.



Source: *Soft prompts (huggingface.co)*

2. Prefix tuning

Prefix tuning closely resembles prompt tuning, as both involve adding task-specific vectors to the input. These vectors are adjusted and updated while the rest of the pre-trained model's parameters remain unchanged.



Source: *Soft prompts (huggingface.co)*

The key distinction is that prefix tuning integrates these parameters across all model layers, unlike prompt tuning, which only modifies the input embeddings. Additionally, prefix parameters are optimized using a separate feed-forward network (FFN) to avoid instability and performance issues. Once the soft prompts are updated, the FFN is removed.

3. P-Tuning

P-Tuning involves adding a trainable embedding tensor to optimize the prompts. This optimization is done with a prompt encoder, typically a bidirectional LSTM.

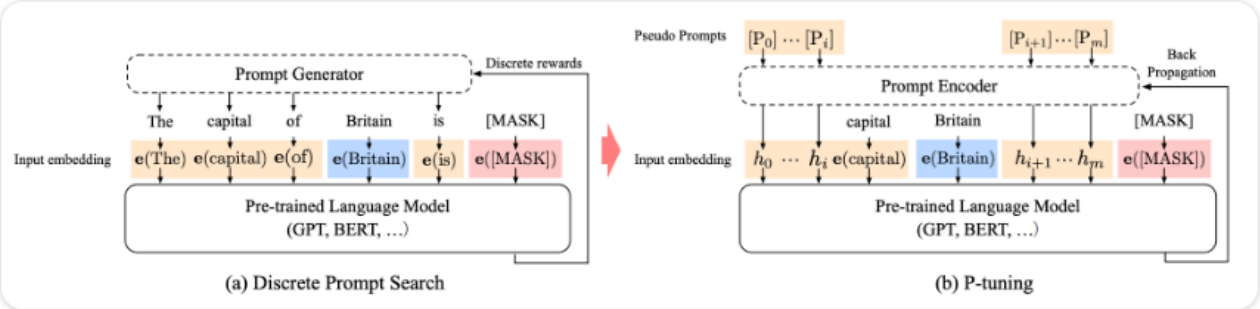
Key points about P-tuning:

Flexible placement: Prompt tokens can be placed anywhere in the input sequence, not just at the beginning.

Input-only addition: Prompt tokens are added only to the input, not to every layer of the model.

Anchor tokens: Adding anchor tokens can enhance performance by highlighting important parts of the input.

P-tuning is generally more effective than manually creating prompts and allows GPT-like models to perform well on tasks typically suited for BERT-like models.



Source: *Soft prompts (huggingface.co)*

Best practices for using soft prompts

There are several best practices to leverage the full potential of soft prompts and their effectiveness.

- First is a robust pre-trained model that acts as a base to ensure that the soft prompts can effectively leverage the existing knowledge.
- The diverse and representative data ensures the data sets are fine-tuned for the tasks for learning and generalizing the soft prompts.
- Evaluating the model's performance regularly and adjusting the soft prompts enhances the model's performance.
- Cross-validation monitors the data overfitting to supervise the model's performance on the unseen data.

Benefits of soft prompts

Let's explore the benefits of soft prompts.

Efficiency: Soft prompts increase task adaptation with fewer resources than full model fine-tuning, optimizing only a small subset of parameters.

Flexibility: Soft prompts fine-tune models for various tasks and help in the dynamic environment where requirements change frequently, such as sentiment analysis to text classification.

Scalability: It is easy to scale soft prompts for different models and data sets, making them a robust solution for diverse NLP applications. This ensures that as the models and data sets evolve, the soft prompts remain a viable and effective tool.

Summary

Soft prompting offers a powerful alternative to traditional model fine-tuning for efficiently adapting LLMs for specific tasks. By leveraging learnable embeddings, it provides a more adaptable and efficient way to enhance model performance across diverse tasks. Understanding methods like prompt-tuning, P-tuning, and prefix tuning opens new possibilities for leveraging AI in dynamic environments. As the field evolves, soft prompting is set to play an increasingly vital role in intelligent system development. Embracing these advanced techniques ensures that NLP applications remain at the forefront of technological innovation, delivering accurate and reliable results.

Author(s)

Raghul Ramesh

Skills Network IBM