Soft Prompting for MoE Models

Claire Bao
Department of Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
claireb@mit.edu

Sabrina Cai
Department of Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
sjcai@mit.edu

Karissa Sanchez
Department of Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
ksanchez@mit.edu

Abstract

Recently there has been a promising new technique called softprompting that aims to recoup the accuracy loss caused by compressing large language models(LLMs). A common technique for LLM models is *prompting* in which a series of tokens called a prompt is appended to the LLM input before entering the model, with the hopes of improving the resulting model output. **Softprompting** builds on this approach by trying to *learn* the most helpful prompt, and a previous paper showed that not only does it have the potential to counteract the loss of accuracy from compressing LLMs, but that these learned prompts can also be transferable across different datasets and tasks, making it a very versatile approach. Our project aims to explore and implement softprompting on mixture-of-experts(MoE) models in the hopes of achieving similar results with this different type of model, which we selected for its unique sparse activation pattern. Although we were limited by compute resources, our preliminary results suggest that it is indeed possible to combine these two modifications to standard LLMs.

1 Background

Our project builds on top of two main approaches : softprompting and Mixture-of-Expert models(more specifically we use the Switch Transformer). In this section, we will explain more about each component as well as important aspects that affect our project.

1.1 Softprompting

The softprompting approach was first proposed in a paper by Xu et al[3]. In this paper, the authors examine how learning prompts to append at the beginning of inputs can improve the accuracy of pre-trained compressed models and recoup some of the accuracy loss from quantization.

The paper first started by loading a LLaMa model trained on the C4 dataset(a natural-language dataset accumulated by collecting language data across millions of web-pages) and compressing the model using pruning and quantization techniques; this served as their "baseline" accuracy. Although we learned in class that pruning and quantizing strategically can limit accuracy losses, this compression of the model still caused a "significant degradation in the quality of generated responses".

Building on top of the pre-trained compressed LLM, they first tried adding a "hard prompt" which was a manually-added, *fixed* prompt that was appended to all of the inputs. This did recoup some of the accuracy, in that the responses started making sense, but it was still occasionally inaccurate and/or incomplete.

Finally, they introduced softprompting, where they trained and optimized the best prompt tokens. Their motivation in doing so was to automate the time-consuming and imprecise method of trying many different prompts and evaluating them all. They used gradient descent, propagated backwards from the final output loss, to update the softprompt tokens.

In the paper's experiments, they used pre-trained LLaMa and OPT models as a base that were compressed using SparseGPT and GPTQ methodologies. The models were then trained using anywhere from 25-100 learnable soft prompt tokens and the model accuracy was evaluated using PPL(or perplexity).

The paper showed that with adding soft prompt learning tokens, a model can recoup accuracy loss caused by the compression of LLMs and can even surpass the original non-compressed model, meaning that it is a promising direction for further exploration in the realm of compressed models.

1.2 MoE

Mixture-of Experts Models The most common structure for mixture-of-experts models is a specific type of model architecture where feedforward networks are replaced with a set of "experts" that contribute their opinion to the final output. Their advantage over standard dense models comes from their sparse activation pattern, which allows the model to have many more parameters without a corresponding increase in computation time.

SwitchTransformer The specific type of MoE model that we will look at is an architecture proposed in a paper by Fedus et al[1] in which each input is routed only to 1 expert rather than the top K. This allows for the capacity of each expert to be decreased since each input is only routed to a single expert and, in general, the computation load is significantly reduced and simplified, both in terms of router computation and implementation. Critically, their study found that this strategy did not decrease model accuracy. These promising results led us to choose this model for our implementation. Specifically, we used the "switch-base-8" implementation of this transformer from HuggingFace.

Our goal We aim to implement these approaches together; in other words, **we aim to add soft prompt learning to the Switch Transformer model in hopes of improving its accuracy on Natural Language problems**. We hope that this will serve as a good baseline from which to pursue various MoE/softprompting projects, as both these methods contribute to improved language model efficiency.

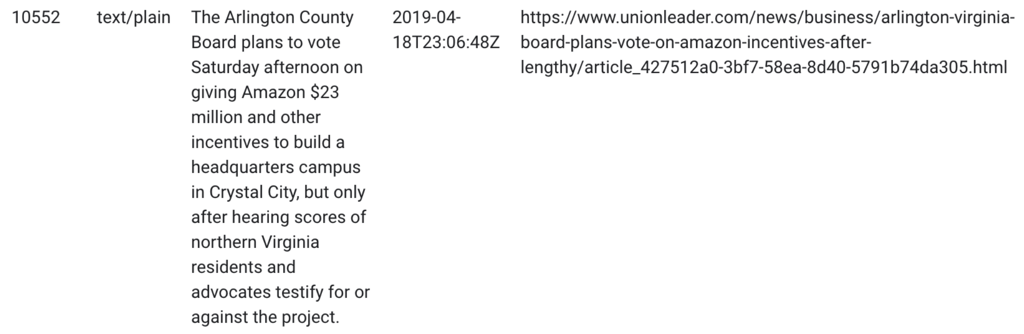
2 Methods

In order to implement this, there were 3 main aspects that we needed to implement/look into : datasets/dataloaders, softprompting infrastructure and interface with the MoE model, and the training and evaluation loops

2.1 Datasets/Dataloaders

We use the Colossal Cleaned Crawled Corpus (C4) dataset, similar to the softprompting paper. The C4 dataset is a dataset is a dataset developed by google which utilizes Common Crawl to scrape the text. Compared to regular web scrapers, the C4 dataset implements heuristics which increase the overall quality of its dataset. Such heuristics include specifications on the number of lines for a text entry and the number of words, to avoid text which is boilerplate and error messages. In addition, it only includes text from webpages which are judged to be at least 99% in English[2]. As a result of these heuristics, the dataset is high quality and better-suited for being trained on NLP tasks in English. We decided to use the C4 dataset because of its quality, in hopes that it would lead to better model performance, and for its ease of access.

The version of C4 we used, hosted by AllenAI, comes in a variety of different flavors, each with varying sizes. Due to disk space constraints on Colab, we decided to use the `realnewslike` dataset,



10552	text/plain	The Arlington County Board plans to vote Saturday afternoon on giving Amazon $23 million and other incentives to build a headquarters campus in Crystal City, but only after hearing scores of northern Virginia residents and advocates testify for or against the project.	2019-04-18T23:06:48Z	https://www.unionleader.com/news/business/arlington-virginia-board-plans-vote-on-amazon-incentives-after-lengthy/article_427512a0-3bf7-58ea-8d40-5791b74da305.html

Figure 1: Abbreviated example of an input from the `realnewslike` variant of the C4 dataset

because it is was the smallest one available at 15GB. As a result, the data we used to train our model was data from news excerpts on the web. One abbreviated example of an input is shown in 1.

In order to load the dataset, we used the dataloader provided by PyTorch. We only used a subset of the training data, since the initial training/validiation split in `realnewslike` was 99.999/0.001. The dataset class we used was the same class used in Xu et al[3], with some modifications to make it compatible with our larger model. Specifically, our dataset class was an extension of the PyTorch dataset class, so that it was compatible with the PyTorch dataloader. Our dataset class used the Switch Transformer's tokenizer to encode all inputs of a dataset at once, then outputted each individual input data point as it was requested by the dataloader. We chose to encode all data points at once to limit the computation time needed as the data was passed through the tokenizer, just as was done in Xu et al[3].

2.2 Softprompting Interface with model

Another main aspect that we needed to approach was how to adapt the different architecture and infrastructure of the MoE model to the appropriate structure of the softprompting approach.

The way the softprompting was handled starts off with taking in the pretrained model, which in our case was the switch transformer "switch-base-8" model from HuggingFace. Then, all of the parameters in this raw model are "frozen", indicating that these inner weights within the model should no longer be adjusted or trained, as we would like our training loop to focus on only learning the soft prompt itself, not adjusting the compressed model.

Soft prompt layers are then added to this frozen SwitchTransformer model. These layers are learned embeddings with the specified number of tokens and the dimensions being of the appropriate hidden size. Once these layers are added they are stored in a named *softprompt* attribute and everything is appropriately initialized.

Then, for each forward pass, the raw input is first passed through this Embedding layer in order to get the learned softprompt; this softprompt is then concatenated with the raw input and this new longer input is passed through the underlying frozen model as appropriate

An interesting challenge that we ran into was that the structure of the underlying model was slightly different between the LLaMA model and the new Switch Transformer model. This entailed us having to write a new wrapper class to appropriately wrap the frozen model with the soft prompting layers to ensure that all calls to the underlying model would work as expected.

Similarly, another challenge that we ran into was that our SwitchTransformer model had a different hidden size than the original models used for softprompting; this required us to reconfigure the appropriate parts in order to take in a flexible hidden size for the model in order to allow it to be more versatile and for it to fit our new model

2.3 Training/Evaluation loop

Model evaluation We evaluated our combined softprompting/MoE model using the standard NLP perplexity formula, to measure how likely our model would be to generate our input sequence.

$$e^{\frac{\sum_{inputs} NLL\_loss}{num\_tokenized\_tokens}}$$

Training Softprompt tokens were initialized to random values and trained following the pattern laid out by Xu et al[3]. Specifically, we used a training loop to minimize the following loss function, where e values represent learnable softprompt tokens and x values represent input tokens. This loss function calculates the negative log likelihood loss of predicting the next token based on all previous tokens.

$$min_E \sum_{t=1}^{n} -\log(Pr_\theta(x_t|e_1, ..., e_k, x_0...x_{t-1})$$

The loss was propagated back to the softprompt tokens to update them.
We also followed the Xu et al[3] paper's optimizer and training hyperparameters (using an AdamW optimizer with a learning rate of 0.5, weight decay of 10^{-5}, etc.)

3 Results

After combining the switch transformer MoE model with softprompting by adding a layer to the frozen switch transformer model, we were able to achieve a baseline perplexitiy from our model. The initial perplexity is **895405**, indicating that there is a high amount of "uncertainty" within our model, but this is not a matter of great concern since it is from our baseline model before training. Unfortunately, the training we were able to complete does not appear to improve perplexity. However, this is likely due to our limited computing resources that prevented us from training on enough examples, so we still believe that this baseline result serves as evidence that this approach can be achieved and is promising.

4 Conclusion

Through this work, we have shown that it is ultimately possible to combine the methods of softprompting and MoE into a unified trainable model. This is an interesting new area of exploration, since the sparse activation pattern of MoE models and the ability of softprompts to improve model performance could together generate an accurate, highly efficient model. Although our initial results do not display greatly improved performance results after training, we still established a promising baseline and structure which can be built upon.

Future work An immediate potential next step for the project would be to try training the model again with higher compute time and memory, to see whether that improves the final model performance.

Another next step could be applying the same softprompting structure to an MoE model that is further compressed or distilled. Although MoE models do already allow for higher numbers of parameters more efficiently than standard dense models, compressed or distilled versions of MoE models could be even more efficient. This could also allow us to see whether the finding that softprompting allows a compressed model to catch up to and surpass the original model's accuracy holds for MoE models as well as dense models.

Further in the future, this project could serve as a baseline for interesting new research directions that make further use of the unique MoE structure. Specifically, one existing problem in MoEs is load balancing between experts. If tokens are not routed equally to all experts, the model will not be fully utilized, making it less efficient and increasing computation time. An interesting research direction might be to see if softprompting can be used to improve MoE load balancing, in addition to simply improving accuracy. However, one challenge from this would be the fact that the most common MoE routing functions are nondifferentiable. This would make it difficult to learn end-to-end the way we currently have our training set up.

Overall, although we ran into obstacles with our limited compute budget, our results may serve as a starting point for interesting future research directions.

References

[1] William Fedus, Barret Zoph, and Noam Shazeer. *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. 2022. arXiv: `2101.03961 [cs.LG]`.

[2] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: `1910.10683 [cs.LG]`.

[3] Zhaozhuo Xu et al. *Compress, Then Prompt: Improving Accuracy-Efficiency Trade-off of LLM Inference with Transferable Prompt*. 2023. arXiv: `2305.11186 [cs.CL]`.