Advanced NLP (M25) - Assignment 3

Deadline: 10th November 2025 23:59

1 General Instructions

Please read and follow the instructions here and mentioned later thoroughly.

1. The assignment must be implemented in Python.
2. You are allowed to use AI tools or any internet resources as long as your implementation is in accordance with the assignment guidelines. You are **not allowed** to **use AI to generate the report**.
3. Please start early as no extension to the announced deadline (10th November 2025) is possible.
4. To ensure a complete understanding of the underlying concepts, please write all required functionality and models (stated below) yourself, without incorporating external libraries or modules.
5. You are required to push the assignment to GitHub Classroom, there will be no moodle link for this assignment.

2 Mixture-of-Experts

Introduction

Models typically reuse the same parameters for all inputs, Mixture of Expert (MoE) models instead select a different set of parameters for different inputs. MoE achieves this by replacing the dense Feed-Forward Network (FFN) layers in a Transformer with a sparse MoE layer. This leads to sparsely activated models but with a higher number of parameters. Sparsity of MoE models is achieved by using only k experts, where k, the number of experts is a hyperparameter.

MoE models also are more complex, have higher communication cost and are unstable during training. Routing algorithms and design improvements are used to reduce the communication and computational cost and also mitigate training instability [1]. Sparsely gated MoE Layers [4] may use feed-forward networks as experts with linear-softmax gating for routing among other alternatives.

You will be implementing a sparse MoE layer from scratch in this assignment along with routing algorithms and a load balancer. Your goal is to replicate a MoE Model for the task of Summarization in English and compare it against the baselines.

Dataset and Task

You will be using the dataset - `EdinburghNLP/xsum` for training and testing your models in this assignment. The dataset contains 204k train samples and 11.3k each for val and test. The task of extreme summarization, introduced in the paper [3], along with the dataset, is to create a

very short, one-sentence summary of a news article that answers the question - "What is the article about?". The new dataset is created from online news articles from British Broadcasting Corporation (BBC).

Baselines

Inference with BART

Run inference using BART (facebook/bart-large-xsum) on your test set. Since the model is already finetuned on xsumm, you would see high scores. This will be considered as one of your baselines that you can compare your model against.

Finetune

Fine-tune/Instruction tune two existing pre-trained model with or without Parameter-Efficient Fine-Tuning [2] on the task of Extreme Summarization. Pick one model (Hugging Face name) each from the two buckets. You may choose one that fits within the GPU you would use. You may use quantized versions of the instruction models if needed. You are **not** allowed to pick already finetuned versions of the following models.

Encoder-Decoder Models

- Pegasus (google/pegasus-large)
- T5 Large (google-t5/t5-large)
- T5 Base (google-t5/t5-base)

Instruction Models

- LLama 1B (meta-llama/Llama-3.2-1B-Instruct)
- LLama 3B (meta-llama/Llama-3.2-3B-Instruct)
- Qwen 3B (Qwen/Qwen2.5-3B-Instruct)

Tasks

1. **Run inference** by loading the pre-trained `BART` model mentioned in Section 2 and the `xsum` dataset from the Hugging Face on the test set.
2. **Finetune** your choice of encoder-decoder model from the list on `xsum` train set and run inference on the test set.
3. **Instruction tune** your choice of instruct model from the list on `xsum` train set and run inference on the test set.

2.1 MoE Transformer from Scratch

Write a Sparse MoE layer from scratch to replace a Feed-Forward Neural Layer in a standard transformer architecture. Implement two routing algorithms viz. `Hash Routing` and Token choice `Top-k Routing`. Also implement a load balancer from scratch. Train the model from scratch on the `xsum` train set.

It is highly encouraged to push your model checkpoints to hugging face hub (push_to_hub function) so that you can continue training your models in case you run out of compute. This will also allow you to run inference on your models on different machines if needed.

Make sure that the total size of the MoE model (including all experts) can fit on the machine you are training on.

Tasks

1. Implement a **Sparse MoE layer** from scratch along with two routing algorithms and a load balancer leading to two models.
2. Train the models on the `xsum` train set.
3. Run inference on the models with the `xsum` test set.
4. Visualize the expert usage over time.

BONUS

Bonus 1 - Load Balancer Loss

Train your MoE models with the two routing algorithms with and without load balancer loss. Report your finding and provide analysis on your results. You are required to push your models to hugging face hub and share the link in your report.

Bonus 2 - Attention from Scratch

Instead of using library implementation of attention block, implement your own Grouped Query Attention module and use that along with the top-k routing algorithm. Report your findings when using your GQA implementation against the library implementation that was used in the assignment. You are required to push your models to hugging face hub and share the link in your report.

Bonus 3 - LoRA/Distributed MoE

Mixture of expert models can be thought to be inherently distributed in nature through expert parallelism. Implement the MoE layer using a distributed framework (e.g., PyTorch Distributed) to ensure that each expert resides on a different GPU (expert parallelism). Please attempt this only if you have access to multiple GPUs which you can use for the assignment.
OR
Implement LoRA for the from scratch to create LoRA-based experts and use that to train your models.

You are required to report your challenges in getting LoRA or distributed MoE to work. If you have working models, push your models to hub and provide a link in the report.

2.2 Analysis, Report, and Submission

Analysis

The analysis is to be provided for all the models (baselines and trained from scratch). Some pointers for analysis include,

- **Lexical Based Metric**: ROUGE-N, ROUGE-1, ROUGE-2, ROUGE-L and BLEU scores for each model.
- **Embedding based Metric**: BERTScore.
- **Document related metric**: Compression Ratio (summary generated/input document).
- **Extractivness**: Percentage of words (after removing functional words like of the etc.) in Summary which overlap Document.

- **LLM-as-judge**: Do Factual Checking using metrics like, SummaC, by comparing the generated summary against the input document.
- **Human Evaluation**: Analyse at least 3 samples from each model yourselves and give summary of your observation based on following dimensions:
  - Content Relevance: How relevant was the content of summary generated. Did it exclude any main points.
  - Coherence: Does summary has smooth flow to form a unified text or was there tangents in terms of semantic flow.
  - Fluency: Grammar wise how was the summary.
  - Facts: Information available in summary is faithful to input document or not. *Note*: A summary can be highly relevant (it covers the main topics of the source) yet be factually inconsistent (it contains verifiable errors relative to the source text).

3 Submission

To ensure your code is version-controlled and backed up, please commit your solutions to the following GitHub Classroom link. When you access the link for the first time, you will see a screen similar to Figure 1. Simply select **your roll number** and proceed to the next step, this will automatically create a repository associated with your GitHub ID. **Please make sure to select your roll number when accepting the assignment. Do not click "Skip", or your repository will be invalidated.** And push the models to HF and keep them ready for

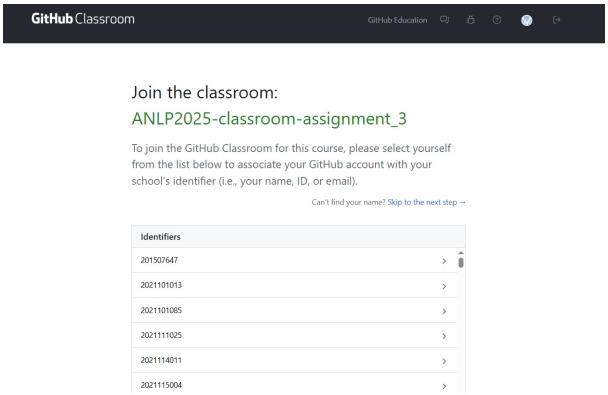

Figure 1: GitHub Classroom invitation screen displayed during first-time setup of the assignment repository.

evaluation.

`report.pdf`: A comprehensive report containing the hyperparameters, compute details and analysis required for your assignment and any bonus you may have attempted. Instructions to run the code and links to your models are also to be included in the report. The report also needs to be pushed to the github classroom.

References

[1] William Fedus, Barret Zoph, and Noam Shazeer. "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity". In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 5232–5270. URL: https://www.jmlr.org/papers/volume23/21-0998/21-0998.pdf.

[2] Sourab Mangrulkar et al. *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods*. https://github.com/huggingface/peft. 2022.

[3] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. "Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen Riloff et al. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1797–1807. DOI: 10.18653/v1/D18-1206. URL: https://aclanthology.org/D18-1206/.

[4] Noam Shazeer et al. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=B1ckMDqlg.