University of Macau

CISC7021 – **Applied Natural Language Processing**

Assignment 1, 2024/2025

(Due date: ***17 September 2024***)

Introduction

In this assignment, we prepare a toy language model using the Llama-2 (Touvron et al., 2023) architecture and evaluate the perplexity. You will also learn how to continue pre-training a base language model using PyTorch[1] and HuggingFace libraries.

Skeleton Code and Environment Setup

All the code is provided to help you get started quickly. You can reuse it or adjust it according to your needs. Necessary tutorials and explanations of function parameters are included in the notebook file.

Pre-training a language model is resource-intensive. To balance effectiveness and computational cost, we focus on a small-scale toy model, ensuring all work can be done using free GPU resources like Colab and Kaggle. Some useful tutorials include:

- https://www.geeksforgeeks.org/how-to-use-gpu-in-google-colab
- https://www.geeksforgeeks.org/how-to-load-a-dataset-from-the-google-drive-to-google-colab
- https://www.geeksforgeeks.org/how-to-use-gpu-in-kaggle/

Note: Kaggle Notebooks offer 30 hours of GPU usage per week. However, you must verify your phone number to access this feature.

You are also welcomed to use your personal GPU cards for this assignment, if available.

Base Language Model

The base language model is a toy model pre-trained on the TinyStories (Eldan et al., 2023) dataset using the Llama-2 architecture. Specifically, this base model was pre-trained on a synthetic dataset of short stories that contain words typically understood by 3 to 4-year-olds. Although this model is much smaller than Llama-2[2] and has only 42 million parameters, it can still produce a diverse set of fluent and consistent stories. We use this model to help you understand the pre-training and generation process of Llama-2 and perform continual pre-training on our dataset.

Note: We tested on Colab and Kaggle to ensure that decoding and continual training can be done using free GPU quotas.

[1] https://pytorch.org/

[2] Even the smallest Llama-2 model has 7 billion parameters.

Data

We provided 11,500 machine-translated Chinese/Portuguese children's stories, which is available at the UMMoodle.[3] The data is split[4] into training (10,000), dev (500) and test sets (1,000). If you would like to use more pre-training samples, you can collect them from other sources (see below).

Tasks

Detailed instructions of each task can be found in the attached notebook file.

1. Use `model.generate()` function to generate English stories using various prompts and decoding settings. Please feel free to explore any interesting phenomena, such as the impact of different prompts and the effects of different decoding algorithms and hyper-parameters. You are encouraged to discuss your findings based on objective evaluations (e.g., text statistics) or subjective case studies.
2. Use `compute_ppl()` function to evaluate the perplexity. Ensure that you evaluate both the English and Chinese (or Portuguese) test data we provided. You are encouraged to collect more diverse text data and discuss your findings regarding the language understanding capacity of the base model.
3. Use `trainer.train()` function to perform continual pre-training on non-English training data (e.g., Chinese or Portuguese). We have implemented data preprocessing and the training pipeline, so you are not required to optimize these components. Instead, focus on tuning the training hyperparameters and observe the changes in model performance. You are encouraged to show the variations of training/validation loss during pre-training process.
4. (Optional) Try your best to collect more story data and improve model performance in the targeted language.

Note: We have prepared Chinese and Portuguese datasets. For continual pre-training, you can also create a dataset in another language that you are proficient in. Useful resources:

- TinyStories dataset: https://huggingface.co/datasets/roneneldan/TinyStories
- Multilingual translation model: https://huggingface.co/facebook/nllb-200-distilled-600M

Submissions

Notebook (20%) Submit a Jupyter Notebook with your code and experiments. Record the results in your environment.

[3] https://ummoodle.um.edu.mo/mod/assign/view.php?id=3689490
[4] https://cs230.stanford.edu/blog/split/

Report (80%) Submit a 2-4 pages report of your work. 1) It should clearly present your experiments, achievements, and solutions to encountered problems. 2) Include tables or graphs of data (e.g., corpora statistics), evaluated perplexities, training curves, etc. 3) Discuss your findings regarding generation, evaluation and model training. The report should follow the ACL proceeding LaTeX format.[5] Do not use Word template!!

Use of Generative AI

Please attach an additional section named "Use of Generative AI" at the end of your report. Whether it is for research, editing or just getting those creative juices flowing, let us know how you have used it in your assignments. At University of Macau, all submissions, written or otherwise, should be accompanied with any or all of the following statements, as appropriate.[6]

Declaration Example:

I acknowledge the use of [name of AI tool(s) and hyperlink] to help me […].
I declare that [content…] generated by AI has been presented and submitted as my own work.

Acknowledgements

The base model checkpoint was converted from llama2.c project.[7] The PPL evaluation code was adapted from the Hugging Face "evaluate" library, and we modify them to avoid causing serious memory issues in the Colab environment.

References

[1] Eldan, R., & Li, Y. (2023). Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*.

[2] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., . . . Bhosale, S. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

[5] https://github.com/acl-org/acl-style-files
[6] https://ctle.um.edu.mo/resource/learning-with-generative-ai-at-um-module/
[7] https://github.com/karpathy/llama2.c