The Chinese University of Hong Kong, Shenzhen
School of Data Science

DDA4220/MDS5122/AIR5011/AIR6011/MBI6011

Assignment 2

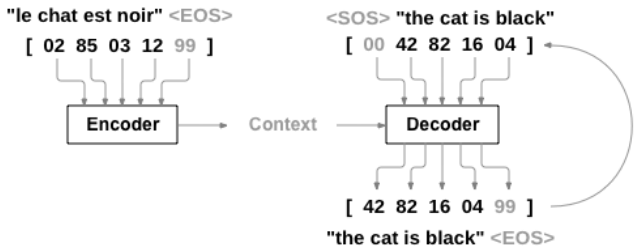
Due by: 23:59, April 30th, 2025

Instructions:

1. You must submit your assignment on Blackboard. Please upload a PDF file along with your code (excluding large files), and ensure that the PDF is not included in the zip file. The file should be renamed to something like **ZhangSan-123456789-hw2.pdf** (your name, student ID, homework ID).
2. Your submission must be clearly answered and well-presented to receive full credit. Please detail how you leverage your acquired knowledge to solve the problem and present your proposed solution step by step. Ensure that your solutions are legible and written in English.
3. The programming language is Python, and your code should include necessary comments so that others can easily follow its logic.
4. If your answers are inspired by other sources (*e.g.*, discussions with classmates, the Internet, or AI), please maintain academic integrity by solving the problems as independently as possible and including a reference or acknowledgment section in your submitted document to reflect how these sources inspired you. Copying answers violates academic integrity.
5. Late submissions or instances of plagiarism will not be graded.

Part A: Chinese-to-English Translation (60 points)

As we have learned in class, Sequence-to-sequence (seq2seq) models have achieved remarkable success in machine translation tasks. These models comprise an encoder that compresses the input sequence into a context vector and a decoder that generates the output sequence from this context vector. The incorporation of attention mechanisms has notably enhanced the model's capacity to align and translate words, enabling the decoder to focus on relevant parts of the input sequence during each decoding step.



In this part, you need to implement a Chinese-to-English neural machine translation system using a PyTorch seq2seq model with attention mechanism. You may borrow some code from here, even though the code was originally intended for French-to-English translation. You can download the dataset from cmn-eng.zip.

Requirements:

1. **Load and process the dataset (15 points).** You need to train on the entire dataset consisting of around 30k sentences. Please randomly partition the dataset into training and testing splits with a ratio of 9:1.

Please note that some functions here may not be directly applicable to our case, so you need to handle this gracefully, *e.g.*, Chinese characters cannot be represented as ASCII codes. You may find the BPETokenizer and GPT2Tokenizer, which convert string sentences into categorical labels, very helpful.

2. **Implement a seq2seq model with attention using PyTorch (5 points).** You can directly copy the code implementation from here.
3. **Train the network starting from random initialization (15 points).** Visualize the training loss curves, display at least 10 examples in a table (including input, translation, and ground truth), and visualize the attention maps of each layer as results for comparison. Your results must demonstrate reasonable accuracy and quality.
4. **Write a minimal GPT network, then train and evaluate it, just as you did with the RNN (20 points).** You may borrow code from this source. You can choose a smaller one to meet your needs.
5. **Ensure that your report is well-presented, the code is reproducible and well-documented with a README file, and that it includes a conclusion section elaborating on what you have learned from the experiments (5 points).**

Part B: Fine-tuning a Quantized LLM With LoRA (40 points)

We have witnessed the power of large language models (LLMs) in recent years. However, training an LLM from scratch is very time-consuming; a more feasible approach to applying the model to a custom scenario is to fine-tune a pretrained LLM, enabling it to acquire domain-specific knowledge. In this experiment, we are required to fine-tune an LLM to help the model master some knowledge of law.

We will use the DISC-Law-SFT dataset, which consists of around 403K data points, utilize LLaMA-Factory as a coding framework for fine-tuning, and leverage Qwen2.5 as the base model. We recommend using its instruction version with a model size of $\leq$ 7B in your experiment; for example, the model name could be "Qwen/Qwen2.5-7B-Instruct". LLaMA-Factory automatically handles Quantized LoRA training for you.

Environment Setup Guidance: First, download the dataset file, and then follow this official instruction to install LLaMA-Factory. It is preferable to have access to a GPU with 24GB of memory; otherwise, it is perfectly fine for you to reduce your model size by selecting a slightly smaller model.

Requirements:

1. **Load and process the dataset (5 points).** You need to leverage the entire dataset, which consists of around 403k examples. You may optionally load and dump the dataset in Alpaca format to facilitate further processing by LLaMA-Factory. We randomly select 90% for training and 10% for testing.
2. **Fine-tune the Qwen2.5 language model (15 points).** You can launch the training from the command line, as stated in this quickstart section. Note that you may need to modify the configuration file, as it was originally intended for the Llama3-8B-Instruct model, not for our Qwen2.5 model.
3. **Write a clear report summarizing your findings (15 points).** Visualize the training loss curves and display at least 5 examples in a table, which should include the question, produced answer, and ground-truth answer. Your results must demonstrate reasonable accuracy and quality.
4. **Ensure that your report is well-presented, the code is reproducible and well-documented with a README file, and that it includes a conclusion section elaborating on what you have learned from the experiments (5 points).**

If you have any issues, please don't hesitate to contact us in our Q&A WeChat group.