NLP Assignment 2

by Guowei Xu

October 27, 2025

Assignment Guidelines

- The assignment is due on **10 p.m. Beijing Time on December 15**. We have a strict late policy: penalties increase to 10%, 20%, 50%, 100% for each day, where the number of days are rounded up.
- The assignment is a **coding assignment**. Submit a zip file containing your code for Part 1 and a report. You will need to submit your **wandb log** for Part 2.
- The full score is **15 points**. Partial credit will be given for partially correct answers. Double-check your submission to prevent typos and compilation errors.
- Collaboration and online search are allowed, but you must acknowledge all collaborators and sources.

This assignment contributes **15 points** to your final grade, distributed as:

- Part 1: 4 points (mandatory)
- Part 2-1: 6 points (mandatory)
- Part 2-2: 5 points (optional)
- Part 3: 5 points (optional)

You must submit either **Part 2-2 or Part 3**, but not both.

Part 1 Pretrain Transformer (4 points)

You'll train a Transformer to perform a task that requires accessing world knowledge—knowledge not provided in the task's training data (especially if generalization beyond the training set is needed). Initially, you'll observe that the Transformer largely fails at the task. Then, you'll learn to pretrain the Transformer on Wikipedia text, which contains relevant world knowledge. Fine-tuning the Transformer on this knowledge-intensive task afterward allows it to leverage the knowledge gained during pretraining. This approach improves the performance significantly.

(a) **Review NameDataset:** Our goal is to predict birthplace based on name, using the `NameDataset` class in `dataset.py`. Run `python dataset.py namedata` to inspect sample data. No code or submission required.

(b) **Construct your model:** Construct a decoder-only model. You may want to complete several coding blocks in `model.py` and `attention.py`. Note that you can try different model architectures, which may help you in the later steps. When implementing the forward pass, use the NLL loss.

(c) **Implement Finetuning (without pretraining) (1 points):** Modify `run.py` for finetuning a Transformer on `NameDataset`. After training, you should report your prediction score on the dev set. You can refer to `run_finetune.sh` and `run_evaluate.sh` for launching details.

(d) **Pretrain:** Implement `getitem()` in `CharCorruptionDataset` for a span corruption task, as per instructions in `dataset.py`. CharCorruptionDataset is a class that generates examples for a simplified span corruption objective used in language model training, and it is explained in detail in the Python document. Debug with the command `python dataset.py charcorruption`. Then you should run `run_pretrain.sh` to get a pretrained model. Your pretrain should finish in less than an hour (in cpu).

(e) **Finetune at pretrained model, and Make predictions (3 points):** Finetune at the pretrained model, and also report your prediction score on dev set. We expect the final performance will exceed 12%. If you encounter difficulties, consider experimenting with alternative architectures and hyperparameters.

Part 2: How to train a large language model (11 points = 6 + 5 optional)

Like in all areas of Artificial Intelligence, when training large language models we often make use of open-source repositories to avoid rewriting common components such as SFT and RL implementations. This part of the assignment aims to develop your ability to utilize and modify open-source frameworks, including environment configuration.

Please note that, in general, library versions often conflict between different repos, so it's recommended to create a separate conda environment for each repo. In addition, some libraries may contain unresolved bugs. If you run into issues, you can first check the repo's issues and PRs to see whether anyone has already mentioned them. Also, when using services such as wandb, Hugging Face, or GitHub, you may encounter network issues. There are several ways to work around them. For example, using wandb's offline mode and uploading later, or using Hugging Face's mirror sites, and so on. If you still can't resolve the problem, please contact the TA in the WeChat group.

In **Part 2-1**, you are required to set up and configure `LLaMA-Factory` (`https://github.com/hiyouga/LLaMA-Factory`) and `Lighteval` (`https://github.com/huggingface/lighteval`). In **Part 2-2**, you will need to configure `verl` (`https://github.com/volcengine/verl`).

For this part, you are not required to submit any code. However, you must provide your training and evaluation `wandb` logs in the form of a `wandb.ai` link.

We recommend using the **Ubuntu** operating system and a GPU with **CUDA** support for this part. Training on other operating systems and GPU architectures is also feasible, as long as the environment is properly configured.

Special Notice for Students Using the Provided AutoDL Computing Resources: Due to the constrained space of the system disk (30G), it is highly recommended to relocate packages and environments to the data disk (default 50G, expandable up to 2T which should be sufficient). Specifically, add the following configuration to your .condarc file:

envs_dirs:

- /root/autodl-tmp/miniconda3

pkgs_dirs:

- /root/autodl-tmp/miniconda3

Additionally, remember to regularly monitor disk usage with the command:

du -sh /path/to/your/directory

Before renting GPUs on AutoDL, be sure to carefully review the guidelines in the repositories to avoid potential CUDA memory issues.

Part 2-0: Use `wandb.ai` for Experiment Logging

`wandb.ai` is one of the most widely used platforms for recording experiment logs. To use `wandb`, you first need to register an account at `https://wandb.ai`, and then obtain an API key from `https://wandb.ai/authorize`. Before starting each experiment, make sure to set the environment variable `WANDB_API_KEY` to the API key you obtained, and set `WANDB_PROJECT` to `NLP2025_<your_student_ID>`.

All three repositories we will use already support automatic logging and uploading of training logs to `wandb`. Each repository provides detailed documentation explaining how to upload training logs to `wandb`, and we will also provide a brief explanation later.

Now, please create a `wandb` project on `https://wandb.ai` named `NLP2025_<your_student_ID>` and set its visibility to **public** or **open**. For example: `https://wandb.ai/xkev/NLP2025_2023011416`.

All of your subsequent experiments must be recorded. Provide the link to this project here.

Part 2-1: SFT (6 points)

In this section, you will improve the performance of the `Qwen2.5-0.5B-Instruct` model (`https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct`) on the `GSM-8K` dataset through supervised fine-tuning (SFT).

(a) **Evaluate the base model on the GSM-8K dataset (1 point):** Use the `lighteval` framework to evaluate the performance of `Qwen2.5-0.5B-Instruct` on the `GSM-8K` dataset. After correctly installing the `lighteval` framework, you can test the model using the following script:

```
WANDB_API_KEY=YOUR_API_KEY \
WANDB_PROJECT=NLP2025_YOUR_STUDENT_ID \
WANDB_NAME=TASK_2_1_1 \
lighteval vllm \
"model_name=Qwen/Qwen2.5-0.5B-Instruct,dtype=bfloat16,
max_model_length=4096,gpu_memory_utilization=0.9,
generation_parameters={max_new_tokens:4096,temperature:0.6,top_p:0.95}"
"lighteval|gsm8k|5"  --wandb --save-details
```

Please make sure to clone the latest version of the `lighteval` repository and **install it from scratch**, otherwise you may encounter unexpected bugs. Please also note that when running the `lighteval` framework, it is normal to encounter missing dependencies. Simply install the required packages manually if this occurs. (for example, **proper versions** of both `vllm` and `wandb` need to be installed manually). Please report your score.

(b) **Finetune the base model (1 point):** Use the `LLaMA-Factory` framework to finetune `Qwen2.5-0.5B-Instruct` on the `alpaca_en_demo` dataset. After correctly installing the `LLaMA-Factory` framework, you can finetune the model using the following script:

```
WANDB_API_KEY=YOUR_API_KEY \
WANDB_PROJECT=NLP2025_YOUR_STUDENT_ID \
WANDB_NAME=TASK_2_1_2 \
FORCE_TORCHRUN=1 llamafactory-cli train \
examples/train_full/llama3_full_sft.yaml
model_name_or_path=Qwen/Qwen2.5-0.5B-Instruct report_to=wandb
dataset=alpaca_en_demo  per_device_train_batch_size=32
```

Please make sure to clone the latest version of the `LLaMA-Factory` repository and **install it from scratch**, otherwise you may encounter unexpected bugs. Please also note that when running the `LLaMA-Factory` framework, it is normal to encounter missing dependencies. Simply install the required packages manually if this occurs. (for example, **proper versions** of both `deepspeed` and `wandb` need to be installed manually). You may need to adjust the hyperparameters according to the length of the training data and the available GPU memory.

(c) **Choose your own training dataset (1 point):** It is not difficult to observe that the example dataset `alpaca_en_demo` is a simple and general-purpose dataset, which has little impact on improving the model's performance on mathematical reasoning datasets such as `GSM8K`. Learning to select appropriate datasets for model training is crucial.

In this part, you are free to choose any public or private dataset to perform supervised fine-tuning (SFT) on the `Qwen2.5-0.5B-Instruct` model (you may also use LoRA fine-tuning). If you're not sure which dataset to choose, you can start with `simplescaling/s1K-1.1`. You may freely modify the training hyperparameters, although the default settings generally yield good results. Please keep the complete logs of all your experiments, and set `WANDB_NAME` to `TASK_2_1_3` for all runs. You may need to adjust the hyperparameters according to the length of the training data and the available GPU memory. Report the method you used to construct your dataset.

(d) **Evaluate your finetuned model (3 point):** In this section, you are required to evaluate the performance of your fine-tuned model on the `GSM8K` dataset. You may adjust the generation parameters as appropriate; it is not necessary to keep them identical to those used for the base model. When reporting to `wandb`, please set `WANDB_PROJECT` to `TASK_2_1_4`. You will find that improving the model's performance may require a large amount of data, and the Qwen series models are already well-trained. Therefore, as long as the fine-tuned model performs better than the base model, the result is considered acceptable.

Part 2-2: RL (5 optional points)

Since Task 2-2 is an optional task, the instructions will not be as detailed as those for Task 2-1, but will instead provide a general outline.

(a) **Reproduce the basic characteristics of RL training (2 points):** `simpleRL-reason` (`https://github.com/hkust-nlp/simpleRL-reason`) is a representative work built using the `verl` framework. This work uses the GRPO algorithm and is among the earliest studies to successfully reproduce the reinforcement learning training paradigm of DeepSeek-R1. Please use the `Qwen2.5-0.5B` model and any training dataset of your choice to reproduce the trends reported in `simpleRL-reason`, where the accuracy increases steadily as training progresses. As a default option, sampling a 1–3K-sized subset from `hkust-nlp/SimpleRL-Zoo-Data` is a good choice. When uploading your results to `wandb`, set `WANDB_NAME` to `TASK_2_2_1`. In your report, please include the corresponding curve.

(b) **Select new RL algorithms and integrate them into `verl` (3 points)**

Please note that when completing this task, you must accept the invitation from GitHub Classroom at `https://classroom.github.com/a/ugFey-8w` , rather than directly cloning the official `verl` repository. Recently, many works in the research community have proposed improvements to RL algorithms such as GRPO, most of which have not yet been merged into the main `verl` repository. Understanding and organizing these works is of significant importance.

Please choose any one such work. As default options, you can try **ERA** (`https://arxiv.org/pdf/2510.08549`) and **VPO** (`https://arxiv.org/pdf/2506.04559`); the former is a purely language-based work, while the latter focuses on multimodal modeling. In your report, briefly describe the improvement in one sentence with necessary formulas, and include the link to your GitHub Classroom repository. You must also include a link to a successfully run `wandb` log.

Please note that the version of verl provided by the open-source code of these methods may differ from the version in the Classroom repository, resulting in some discrepancies in the API. You should try to minimize modifications and ensure that the normal operation of other algorithms in the repository is not affected.

When completing this assignment, always remember that you are merging a new algorithm into verl's official repo, and you should make it easy for users to use your new algorithm. A good way to start is to **create a new folder named after your chosen algorithm** inside the **recipe/** directory.

The TA will carefully review each student's submitted code. Correct implementation of the algorithm and clean, well-organized code are each worth 1.5 points.

You may choose to actually merge one algorithm into the official `verl` main repository (which is indeed possible and will not take much additional time!).

Part 3: Small Project (5 points, Optional)

For this task, you are required to choose a recent research paper in the field of NLP. You will reproduce the results from the paper (so we suggest you to choose a paper with public code), conduct a brief analysis of the approach, and suggest possible improvements. **Please note again that Part 3 and Part 2-2 are mutually exclusive. You should complete only one of them, not both.**

Guidelines

(a) **Paper Selection:** We have already provided a list of NLP paper at the first lecture. You can also choose any other NLP paper.

(b) **Reproduce:** Replicate key results from the paper. **Reproduction is not a trivial work, no one runs code perfectly on the first try.** So document errors you encounter and your resolving process, as this will be a great part in the evaluation. The results you reproduce may differ slightly from those reported in the paper. As long as the discrepancy is within a reasonable range, that's fine. Also, the paper might have used models that are difficult for you to run; in such cases, you can reasonably substitute them with other available models. The baselines in the paper do not need to be reproduced.

(c) **Analysis:** Provide a concise analysis that covers the following:

- **Strengths:** What aspects of the paper's approach or results stand out?
- **Weaknesses:** What limitations or areas of improvement do you observe?

You need experiments to justify your argument.

(d) **Proposed Improvements:** Suggest potential modifications or alternative methods that could enhance the results or address identified limitations.

This small project is independent of the larger project and will be graded separately. You may select a paper related to your large project, but it is not required.Collaboration in terms of discussion among group members is encouraged, but coding and experimental work must be completed individually.

Your report for Part3 should be no more than 2 pages and must include:

- **Core Idea and Main Contributions:** Summarize the main idea and key contributions of the selected paper.
- **Reproduction Challenges:** Procedure to reproduce experiment results. Describe any issues you encountered while reproducing the paper's results and how you resolved them.
- **Experimentation:** Include one (or more) experiment that supports your analysis, either by highlighting a strength or identifying a limitation of the paper's approach.

Please ensure your submission is clear and concise.

Submission Requirement

Your submission should be organized in the following structure:

```
submission
|--  report.pdf
|--  part1
```

Acknowledgement

Part 1 is inspired by CS224N: Natural Language Processing with Deep Learning (Stanford University) and was adapted from last year's assignment.