Lab 1 manual: Observing GPT-2

CSSE 313, Fall 2026

A step-by-step manual for Lab 1: set up your computer, run three short experiments on GPT-2 small and large, and write up what you find. Here is the starter code package.

Overview

In Lab 1 you study GPT-2 from the outside: what its parts look like, where its answers form, and which attention heads do what. You run provided scripts on GPT-2 small (124M parameters) and GPT-2 large (774M), write four short functions, and write a lab report.

Allow about 6.5 hours of work, plus 30 to 60 minutes of setup the first time.

You hand in a lab report of 2 to 3 pages, your scripts, and the `results/` folder your scripts produced. The report must stand on its own: every figure, table, number and prediction it relies on appears in the report itself. The scripts and results are only there so your instructor can verify the report.

Setup (do once)

This takes 30 to 60 minutes, mostly waiting for downloads. You need about 4 GB of free disk space. The labs were tested with Python 3.13, `torch` 2.7.1 and `transformers` 5.12.1.

Step	Mac	Windows
1. Open a terminal	Terminal (Applications > Utilities > Terminal)	PowerShell (Start menu, type "PowerShell")
2. Check your Python version (3.13 recommended)	`python3 --version`	`py --version`
3. Unzip the folder	Double-click the zip file in Finder	Right-click the zip file > Extract All
4. Move into the folder	Type `cd ` (with a space), drag the folder into the Terminal window, press Return	Open the folder in File Explorer, click the address bar, type `powershell`, press Enter. This opens PowerShell inside the folder
5. Create a virtual environment (once)	`python3 -m venv .venv`	`py -m venv .venv`
6. Activate it (every session)	`source .venv/bin/activate`	`.venv\Scripts\Activate.ps1`
7. Install the packages (once)	`pip install -r requirements.txt`	`pip install -r requirements.txt`

Step	Mac	Windows
8. Check the setup (once per model)	`python check_setup.py` then `python check_setup.py --model large`	`python check_setup.py` then `python check_setup.py --model large`

Notes:

- If Python is missing or older than 3.10, install Python 3.13 from python.org. On Windows, tick "Add python.exe to PATH" in the installer.
- On Windows, if step 6 says that running scripts is disabled, run `Set-ExecutionPolicy -Scope Process RemoteSigned` and try step 6 again. Or use Command Prompt instead of PowerShell and activate with `.venv\Scripts\activate.bat`.
- After step 6 your prompt starts with `(.venv)`. If it does not, the environment is not active.
- If you use Anaconda or Miniforge instead, the same on both systems: `conda create -n gpt2lab python=3.13`, then `conda activate gpt2lab`, then step 7.
- The first run of `check_setup.py` downloads the model (about 0.5 GB for small and 3 GB for large) into `.cache/huggingface` in your home folder. It downloads only once.

Each time you start working

Repeat these two steps in every new terminal window. Once the environment is active, every command in this manual is the same on both systems, and `python` is the right command on both (not `python3` or `py`).

Step	Mac	Windows
1. Open a terminal in the folder	Terminal, then `cd` to the folder	PowerShell opened in the folder (setup step 4)
2. Activate the environment	`source .venv/bin/activate`	`.venv\Scripts\Activate.ps1`

What to do

Work through the three parts in order. In each part, run the script for both models, small and large. Everything the scripts save goes into the `results/` folder.

Part 1: Orientation and anatomy (about 2.5 hours)

- **Run:** `python part1_anatomy.py --model small`, then the same with `--model large`.
- **Write:** open `part1_anatomy.py`. Replace the five strings in `STRINGS` with strings of your own (include a number, a rare word, and one word with and without a leading space). Write the three functions marked TODO: `predicted_params`, `reconstructed_residual` and `attention_checks`. The script runs without them and tells you what is missing.
- **In your report:** the tokenization notes and the leading-space convention; your parameter formula and how it compares with each model's count, with any difference explained; the table of activation shapes with every dimension labelled; the residual-sum check (the maximum error) and what it means; the attention check.

Part 2: Logit lens (about 2 hours)

- **Write:** five prompts of your own in `data/my_prompts.json` (format below), and the function marked TODO, `first_top1_layer`, in `part2_logit_lens.py`.
- **Run:** `python part2_logit_lens.py --model small`, then `--model large`. After both have run, the script also draws the two models on the same axes.
- **In your report:** the figure; a table of the first layer at which the correct token is top-1 and the layer from which it stays top-1; one prompt where the correct token is top-1 at some layer and then lost; one limitation of the logit lens, with an example from your data; and your answer to this question: does the larger model resolve the answer earlier in relative depth, in absolute depth, or neither?

Part 3: Attention heads (about 2 hours)

- **Run:** `python part3_heads.py --model small`, then `--model large`. Add `--top 10` to list more heads. The first time, step 5 is skipped, and the script says so.
- **Write your prediction:** read the top induction head the script reports. In `predictions_part3.md`, predict how that head will behave on the sentence in `data/part3_name_sentence.txt`, in which a name appears twice. Then run the script again so that step 5 runs.
- **In your report:** the heatmaps; the loss on the first and second copy of the random tokens, and why the second is lower; the top induction heads and their attention patterns described in words; the layers of the top previous-token and induction heads in each model, and whether they fit the idea that induction needs an earlier head; your prediction, quoted word for word, and what happened; and half a page on what attention patterns alone can and cannot establish.

Files you create yourself

Create these in a plain-text editor and save them in the folder shown, with exactly these names.

File	Where	When
`data/my_prompts.json`	the `data` folder	Part 2: your own five prompts
`predictions_part3.md`	the main folder, next to the scripts	Part 3: before the natural-text check

	Mac	Windows
Editor	TextEdit (choose Format > Make Plain Text before saving), or VS Code	Notepad, or VS Code
Watch out for	TextEdit saves rich text by default, which breaks `.json` files	Notepad may add `.txt` to the name. In the Save dialog set "Save as type" to "All files"

The format of `data/my_prompts.json` is a list. Each prompt has an answer that starts with a space and is a single token. The script skips any prompt whose answer is not a single token and tells you why.

```
[
  {"prompt": "The capital of Norway is", "answer": " Oslo"},
  {"prompt": "The opposite of fast is", "answer": " slow"}
]
```

The report

The report is a traditional lab report of 2 to 3 pages, not counting figures. Start from `report_template_lab1.md`. It has these parts:

- **Introduction and hypotheses:** the question each part asks and what you expect to find. Quote your Part 3 prediction here.
- **Methods:** models and library versions, the prompts you used, the exact commands in the order you ran them, and a statement about your use of AI (below).
- **Results:** the figures and tables, embedded in the report, each with a caption and the name of the file that produced it.
- **Discussion:** what the results show, what they do not show, and where small and large differ.
- **Limitations and open questions.**

The report must stand on its own. Every figure, table, number and prediction it relies on appears in the report itself. Your scripts and the `results/` folder are submitted only so that your instructor can verify it.

Use of AI assistants. You may use an AI assistant only to have things explained to you: a concept, a function in `gpt2lab`, or an error message. Do not use one to write or fix code, to write or edit any part of your report, or to draw conclusions from your results. State in Methods that you followed this policy, and name any assistant you used for explanations.

If something goes wrong

Symptom	Mac	Windows
`python: command not found`, or `'python' is not recognized`	Activate the environment (setup step 6), or use `python3`	Activate the environment (setup step 6), or use `py`. If neither works, reinstall Python with "Add python.exe to PATH" ticked
`No module named 'torch'`	The environment is not active. Activate it (setup step 6)	Same
"Running scripts is disabled" when activating	Does not happen on Mac	Run `Set-ExecutionPolicy -Scope Process RemoteSigned`, or use Command Prompt (see the setup notes)
`Could not find a version`	Intel Macs cannot install this version of	Check that you have

Symptom	Mac	Windows
`that satisfies the requirement torch==2.7.1`	PyTorch. Ask your instructor for another option	Python 3.13 (setup step 2). If you do, ask your instructor
A download stops or is slow	Run the command again. The download resumes	Same
The large model is killed or the computer freezes	Close other programs and try again. Run small first	Same
`[not done yet]` in the output	A function marked TODO is not written yet. This is expected until you write it	Same
Step 5 of Part 3 is skipped	Create `predictions_part3.md` in the main folder, then run again	Same
Odd errors after you changed a script	Delete the `results` folder and run again	Same

To look at your figures: on a Mac, `open results` in Terminal (or use Finder); on Windows, `explorer results` in PowerShell (or use File Explorer).