

HOMWORK 0^{*}

COURSE PRIMER[†]

24-880 AI AGENTS FOR ENGINEERS
<http://cmuagents.github.io>

TAs: Peter Pak

Instructions

- **Collaboration Policy:** Please read the collaboration policy in the syllabus.
- **Late Submission Policy:** See the late submission policy in the syllabus.
- **Submitting your work:** You will use Gradescope to submit answers to all questions and code.
 - **Written:** You will submit your completed homework as a PDF to Gradescope. Please use the provided template for the submissions which can be written in $\text{\LaTeX}$ (via [Overleaf](#)) or directly on the PDF document. Each answer should be within the box provided. If you do not follow the template, your assignment may not be graded correctly and there will be a **2% penalty** (e.g., if the homework is out of 100 points, 2 points will be deducted from your final score).
 - **Programming:** This first assignment follows an already existing jupyter notebook content, no code will need to be submitted.
- **Materials:** The data that you will need in order to complete this assignment is posted along with the writeup and template on the course website.

Question	Points
Getting Started	15
Large Language Model Refresher	80
Collaboration and Generative AI Questions	5
Total:	100

^{*}Adapted from 10-623 Generative AI Homework Template

[†]Compiled on Tuesday 13th January, 2026 at 19:37

Introduction

This assignments will provide be a refresher for several of the topics covered within this course. Along with this, it will help onboard you to some of the tools and packages we will use throughout the semester.

At a high-level, you will proceed as follows:

1. Read the documentation for `uv` and install onto your local system.
2. Create a HuggingFace account
3. Follow instructions for LLM from scratch tutorial

Computing Environment

First you need to setup your computing environment, this is useful for future homework assignments. Below we outline how you could do so on your machine.

Local Environment

For this course, we suggest using the `uv` project and package manager as it provides a lightweight solution to automatically handle many of the dependency and python versioning challenges encountered with setups such as Anaconda, MiniConda, etc. If you're familiar with tools such as `pip` and `venv`, then `uv` will be pretty similar to what you're used to.

You can explore all of the available commands using `uv --help` but in short, here are a couple of useful commands that you'll often use:

<code>source .venv/bin/activate</code>	(Linux and Mac) Activates python environment.
<code>source .venv/Scripts/Activate</code>	(Windows) Activates python environment.
<code>uv sync</code>	Installs / synchronizes projects dependencies.
<code>uv add <package></code>	Installs <code>pip</code> package and adds it to project dependencies.
<code>uv remove <package></code>	Removes <code>pip</code> package dependency from project.
<code>uv run <script.py></code>	Runs python script inside of <code>uv</code> environment.

1. Follow the instructions linked below to install UV.

<https://docs.astral.sh/uv/>

2. Navigate to folder `homework-0-course-primer/` and synchronize dependencies with:

```
uv sync
```

3. Activate the environment

Mac and Linux:

```
source .venv/bin/activate
```

Windows:

```
source .venv/Scripts/activate
```

4. Test `uv` works and run `main.py` file.

```
uv run main.py
```

1 Getting Started (15 points)

1.1. (5 points) Did you read the syllabus?

☐ Yes

☐ No

1.2. (5 points) Create or provide the username to your HuggingFace account

1.3. (5 points) Did you successfully install `uv` onto your machine?

☐ Yes

☐ No

2 Large Language Model Refresher (80 points)

In this section, you will create a Large Language Model (LLM) from “scratch” following the instruction set from HuggingFace’s *Training a Causal Language Model from Scratch* tutorial.

<https://huggingface.co/learn/llm-course/en/chapter7/6>

We will follow most of the steps provided in this tutorial as it guides us on how to use HuggingFace’s abstractions for dataset and model management. In addition, a couple more tasks will help refresh us on some of the basic concepts of how an LLM is trained and inferenced upon. Since most of the code is already provided to us, we’ll look deeper into what’s going on under the hood.

Dataset

Let’s take a look at the `codeparrot` dataset hosted on HuggingFace

<https://huggingface.co/datasets/transformersbook/codeparrot>

- 2.1. (5 points) Briefly describe this dataset. What kind of data is in this dataset where does the majority of its content come from?

- 2.2. It seems that this dataset consists of a large amount of data.

- 2.2.a. (1 point) What do the authors of this dataset claim its uncompressed size to be?

- 2.2.b. (1 point) What is its compressed size?

- 2.2.c. (1 point) How many files?

- 2.2.d. (5 points) How was this dataset obtained?

Let’s go a bit deeper than just saying Google BigQuery

2.3. Now, let's look at the paper *Scaling Laws for Neural Language Models* by [Kaplan et al. \[2020\]](#)

2.3.a. (5 points) What relationship do the authors observe between model size and language modeling performance, and how is this relationship characterized mathematically?

2.3.b. (5 points) What do the authors mean by *compute-optimal* training, and why does it suggest stopping training before full convergence?

2.3.c. (10 points) Suppose you are training a code language model using the CodeParrot dataset. Using insights from the scaling laws paper, discuss how you would decide on an appropriate model size and amount of training data given a fixed compute budget.

Tokenization

For this section we'll look at the tokenization for processing the dataset along with some concepts from the reading *Attention is all you need* [Vaswani et al. \[2023\]](#).

2.4. (5 points) Briefly describe how the dataset is tokenized and prepared for training in this portion of the tutorial. Why split documents into fixed-length chunks instead of truncating them?

2.5. The tutorial fixes a relatively small context size and applies chunking during tokenization.

2.5.a. (2 points) What context length (in tokens) is used for training in this tutorial?

2.5.b. (2 points) When tokenizing the first two examples, how many total chunks are produced?

2.5.c. (5 points) Why are chunks that are shorter than the maximum context length discarded during preprocessing?

2.6. (5 points) The tutorial notes that increasing the context length improves the amount of information available to the model but increases computational cost. Explain how the choice of context length affects both training efficiency and memory usage, and relate this tradeoff to the self-attention mechanism introduced in *Attention Is All You Need* [Vaswani et al. \[2023\]](#).

Model Initialization

2.7. (5 points) Briefly describe how the model is initialized in this tutorial. What does it mean to initialize a model *from configuration* rather than using pretrained weights?

2.8. The model configuration directly determines the size and capacity of the network.

2.8.a. (2 points) What is the total number of trainable parameters in the initialized GPT-2 model?

2.8.b. (5 points) The tutorial uses the same architecture as the small GPT-2 model but trains it from scratch on a much smaller dataset. Using insights from *Scaling Laws for Neural Language Models* [Kaplan et al. \[2020\]](#), discuss how model size and dataset size together influence expected performance in this setting.

Training and Evaluation

- 2.9. (5 points) Briefly describe the training setup used in this tutorial. How does the Trainer (or Accelerate-based loop) construct batches, compute the loss, and update the model parameters?

- 2.10. Several training hyperparameters control the effective batch size and optimization behavior.

- 2.10.a. (3 points) What is the effective batch size used during training, and how is it computed from the provided hyperparameters?

- 2.10.b. (3 points) What learning rate schedule is used, and what is the purpose of the warmup phase?

- 2.10.c. (5 points) During training, model performance is evaluated using loss and perplexity. Explain what perplexity measures in the context of language modeling, and relate its behavior during training to the empirical scaling trends discussed in [Kaplan et al. \[2020\]](#).

3 Collaboration and Generative AI Questions (5 points)

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found in the syllabus.

- 3.1. (2 points) Did you collaborate with anyone on this assignment? If so, list their name or Andrew ID and which problems you worked together on.

- 3.2. (3 points) Did you use generative AI for parts of this assignment? If so, include the input prompts along with links to the conversation.

References

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models. January 2020. doi: 10.48550/arXiv.2001.08361. URL <http://arxiv.org/abs/2001.08361>. arXiv:2001.08361 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. URL <http://arxiv.org/abs/1706.03762>. arXiv:1706.03762 [cs].