Homework 1

11-667 Fall 2025

Due date: 09/11/2025 11:59 PM EST

In this homework, you will investigate existing pre-training datasets. Then, you will replicate a data pre-processing pipeline that converts raw web-crawled data into a cleaned dataset that is easy and efficient to train on. Through this homework, you will explore (a) attributes of large scale pre-training datasets and their properties [*Problem 1*] (b) different steps involved in cleaning, reducing, diversifying [*Problem 2.1-2.2*] and de-duplicating pretraining data [*Problem 2.3*] and finally, (d) converting it into a standardized format for improved use in pre-training [*Problem 2.4*]. In your responses, be precise and on point. Concise answers will be considered superior to lengthy ones if they convey similar information.

Instructions for Submission: Complete the coding portions in `mini_ccc.py` and `homework.py` and write the answers to all the accompanying deliverables in a file that will be submitted as `report.pdf` (we recommend using the provided template). Make sure that you individually select each file and then compress them to create a single zip folder for your submission on Gradescope.

Note: *This homework requires about 3GB of disk space to run.*

Problem 1: Understanding Training Data

[Question 1.1] (***Written, 5 points***) Pick two datasets from the following and answer the questions below for each of them.

- Dolma
- C4
- Project Gutenberg
- The Pile
- RedPajama V1
- The Stack

DELIVERABLES FOR Q1.1

For your two chosen datasets, answer the following questions. ***Please answer as concisely as possible (2-3 lines).***

A. When is the "knowledge cutoff" for your chosen dataset, that is, the latest creation date of any document in the dataset.

B. Where was the data sourced from? Does the dataset consist primarily of a single type of content or does it have multiple types?

C. Identify an open-sourced language model that was trained entirely or in part on this dataset.

D. What license was the dataset released under? Do all documents in the dataset have the same copyright protections (or lack thereof)? If yes, describe these protections *briefly*. If no, give an example of two portions of the dataset that may have different protections.

E. Name a task that a language model trained solely on this dataset might perform poorly at, even with finetuning. Give reasons to support your answer.

[Question 1.2] (***Written, 3 points***) Mitchell et al. (2019) proposed the idea of Model Cards—documentation accompanying every released machine learning model that details its performance and training characteristics. Though nearly all state-of-the-art large language models release Model Cards, the amount of information present in them can vary substantially. Locate the Model Cards for three language models, one that meets each of the following descrpiptions.

- A language model that is only accessible via an API; it does not have public weights.
- A language model that has publicly accessible weights, but the pre-training dataset is not publicly available.
- A language model that has publicly accessible weights and a publicly accessible pre-training dataset.

Then, answer the questions based on the Model Cards you located.

DELIVERABLES FOR Q1.2

Concisely answer the following questions.

A. What are the three models you have chosen?

B. Compare and contrast what the three Model Cards say about the data they are trained on and how it was processed.

C. Discuss a language model use case where you might opt for a language model trained on publicly accessible data, even if its performance is worse than a closed source model.

Problem 2: Training Data Preprocessing Pipeline

The Common Crawl is a public index of the internet. It has been used as a source of internet data for nearly all open-source language models. However, language model trainers first have to take the raw data released by Common Crawl and process it into training-ready datasets. For example, to train T5, Common Crawl was processed into the C4 dataset, and to train OLMo, it was processed into the Dolma dataset. In this problem, you will implement a data processing pipeline for Common Crawl data. You will start with the raw files released by Common Crawl and end up with cleaned text in a dataset format that is ready to be trained on.

[Question 2.1] (***Coding, 3 points; written, 2 points***) Common Crawl data is split across three files with the following formats:

- The Web ARChive (WARC, for short) format is used to store the HTTP responses for urls included in the crawl.
- The WARC Encapsulated Text (WET, for short) format contains plaintext extracted from each webpage.
- The Web Archive Transformation (WAT, for short) format is used to store metadata around each request.

The CommonCrawl is broken up into thousands of *slices* (shards), each of which contains a small fraction of the total data. We have provided you `download_data.sh`, which downloads the three file formats for a single shard of a 2018 CommonCrawl dump. Your task is to start from the WARC file and implement a data processing pipeline that turns the raw HTML dump into formatted and cleaned text that is ready for model training. You are already given the `read_warc_file` function. This function reads the warc file you downloaded and returns a dictionary containing a URL as the key and the raw HTML as its value. After reviewing the function, make the following modifications to `homework.py`.

1. Implement the `html_to_text` function. This function should take as input the raw HTML and output plaintext without any HTML tags. You are allowed to use any of the packages listed below. If you would like to use another package not listed here, please write a Piazza post with your request, explaining what your desired package does that is different than these ones.
   - `html2text`
   - `nltk`

- `BeautifulSoup`

2. Modify the code in `__main__` in order to answer the questions below. Your code for this block will not be graded.

DELIVERABLES FOR Q2.1

Upload your `homework.py` and `mini_ccc.py` to GradeScope to check if the relevant tests for the question are passed. We may also run additional tests of your code.

Answer the following questions. ***Please answer as concisely as possible (max 3 sentences).***

A. How many HTML pages are in this shard of the crawl?

B. How does your `html_to_text` handle web pages which contain HTML tables as part of the content? What about websites which contain code (see example)?

C. How does your `html_to_text` handle the following HTML components: headers (e.g. `h1`, `h2`, etc.), and embedded images (e.g. the `img` tag)?

D. Inspect the WET file either by running `less data.wet` or by using the `read_wet_file` in `utils.py`. Based on your observations, discuss some ways the html-to-text conversion implemented by the CommonCrawl team differs from what you implemented. Which version do you think makes for better language model training data?

[Question 2.2] (***Coding, 8 points; written 4 points***) For the remainder of this homework, we will refer to the web-pages that have been converted to plaintext as *documents*. It is typical to filter training datasets to only contain "high quality" text. Filtering can either be at the document level (removing entire documents which don't meet the quality bar) or it can be applied within each document (—)removing or altering substrings within the text). In class we have discussed several ways of measuring quality, including using an automatic classifiers, heuristics such as document length, and selecting content from certain domains only. In this question, you should implement a method for filtering out individual lines and entire documents if they do not abide by simple conditions designed to judge their likelihood of being useful text. You can read more about the motivation of these conditions in Gopher and RedPajama. You will also be implementing a method to mask Personal Identifiable Information in order to limit unintended disclosure of personal information by language models trained on your data. You will continue modifying `homework.py` and implement the following methods:

- `heuristic_quality_filter`: This method should return `True` iff the document

  - contains no words from the bad words list located here. For convenience, we've included this file in the homework starter code, as well as a function in `utils.py` to read it in.

  - contains punctuation (use `strings.punctuation` for this).

  - contains non-whitespace characters.

  - at least 80% of characters in the document are one of: alphanumeric, punctuation, whitespace.

- `remove_pii`: This method should modify the input string to remove selected personal identifiable information. Note that this is only a toy method that will remove some obvious examples of PII. In practice, much more sophistication would be needed.

  - Transform any occurrences of a US social security number of the form XXX-XX-XXXX, replacing all digits with the letter X.

  - Transform any 10-digit phone numbers (a 10 digit number preceeded by a +1) by replacing all digits with the letter X.

- `clean_text`: This method should remove low-quality portions of the text by performing the following operations. For the purpose of this question, let's define a paragraph as a piece of text separated from other text by a newline, that is: `paragraphs = text.split("\n")`

- Remove any paragraphs that contain more than 100 alphanumeric characters with no whitespace between them.

- Remove any paragraphs that do not contain punctuation.

DELIVERABLES FOR Q2.2

Upload your `homework.py` and `mini_ccc.py` to GradeScope to check if the relevant tests for the question are passed. We may also run additional tests of your code.

Answer the following questions. ***Please answer as concisely as possible (max 3 sentences).***

A. How many documents were deleted in total? Use this to estimate the percentage of the CommonCrawl which would be considered low-quality according to your classifier. Assume there are 3 billion documents in the Common Crawl.

B. Identify two documents you perceive as low-quality which was not caught by your filter. Would it be possible to design heuristic-based filter for documents like these? Do you think it would be more or less effective to train a classifier to identify similarly problematic documents? Explain your reasoning.

C. How do the filtering and cleaning methods you have implemented impact the inclusion of non-English text in the final dataset?

D. Albalak et al. (2024) describe a variety of methods for data filtering. You've already implemented some stages discussed in the paper. Identify and discuss a data cleaning stage that you've not implemented yet that is known to improve training efficiency by reducing data redundancy.

[Question 2.3] (***Coding, 5 points; written 2 points***) In this problem, you will implement a crucial data preparation step, *de-duplication*, on a toy synthetic dataset. De-duplication's broad goal is to reduce a dataset's size by reducing redundant content in the data while preserving the content diversity as much as possible. Keeping this in mind, implement your own de-duplication technique (you can check Albalak et al. (2024) for tips). You are given `topic_dataset.json` which contains some English sentences. Complete the `dedudplicate_texts` function in `homework.py`. Your solution will be judged against a naive string-matching baseline which removes approximately 20 sentences from the current dataset.

DELIVERABLES FOR Q2.3

Upload your `homework.py` and `mini_ccc.py` to GradeScope to check if the relevant tests for the question are passed. We may also run additional tests of your code.

Answer the following questions. ***Please answer as concisely as possible (max 3 sentences).***

A. The term de-duplication is also used in traditional computer science for improving storage efficiency in file systems. Consider a scenario, where a student took the `deduplicate_texts` function you implemented and replaced their traditional file system's de-duplication function with your function. Do you expect to see any consequences? Why? Assume that the traditional file system de-duplication system is a perfect string-matching algorithm.

B. Can you think of any ***disadvantages*** of de-duplication? Explain your answer (1-2 lines) by citing empirical evidence from any peer-reviewed paper of your choice. (ArXiv versions are acceptable).

[Question 2.4] (***Coding, 6 points; written 3 points***) The HuggingFace Datasets library is a popular framework for distributing datasets that can easily be used for model training. In this question, you will convert the dataset you created into the HuggingFace Datasets format. You will find the documentation useful.

You are given `mini_ccc.py`. Make the following modifications to it.

1. Implement the `_info` function. Your implementation should return a `DatasetsInfo` object with a `description` of the dataset and a list of `features` passed in as arguments.

2. Implement the `split_generators()` function. This function should download the .warc file using the `dl_manager` and label it as a "train" split for your data. Try and be as consistent with the intended nomenclature of splits as described in the huggingface documentation for Splits.

3. Implement the function `generate_example()`. This function should read your `.warc` file and yield a tuple containing a url and its corresponding processed plaintext. Note that you **should use** the functions implemented in `homework.py` and `utils.py` for reading and processing the `.warc` file. No need to apply deduplication. The output of this function should be the cleaned documents which pass the quality filter.

DELIVERABLES FOR Q2.4

Answer the following questions. ***Please answer as concisely as possible (max 3 sentences).***

A. How long in seconds does your code take to process the entire dataset. Estimate how long it would take your code to process the entire CommonCrawl (assuming 3 billion documents).

B. Discuss how you could modify your code to make it significantly faster.

C. Tulu is a suite of models trained with a combination of supervised finetuning datasets. Load the dataset used to train this suite of models and observe some of its samples: (a) Can you observe any difference in the way these samples are formatted versus the way you have prepared your dataset? Name this style of formatting and list one advantage for this style. (b) If you were to convert your dataset into this format, would you apply it to both the training and validation sets?

   Hint: SFTTrainer in huggingface return constant length chunks of tokens from a stream of your dataset's examples. Think about when this could be useful!

Upload your `mini_ccc.py` to GradeScope and ensure that the tests are passed. We may also run additional tests of your code.

Problem 3: Use of Generative AI

If you used Generative AI for any part of your homework, you should fill out this question. Failure to do so is an academic offense and will result in a failing grade on the homework. You may omit this question from your submission if you did not use Generative AI.

[Question 3.1] (***Written, 0 points***) Did you use a coding assistant (e.g. GitHub Copilot) that was built into your code editor? If yes, which one did you use? Describe what parts of the code you wrote yourself and which parts it wrote for you.

[Question 3.2] (***Written, 0 points***) Did you converse with a chatbot agent (e.g. Gemini, Claude, or ChatGPT) to help with either the coding or conceptual questions on this homework? If yes, include a table that contains the following information:

- The prompt you passed to the agent. This can be satisfied by providing a link.
- The agent's output for that prompt. This can be satisfied by providing a link.
- A brief description (~1 sentence) of which homework question(s) you used this output for, and how it was incorporated into your final answer.

[Question 3.3] (***Written, 0 points***) Did you have any other use of Generative AI that you would like to disclose?