Carnegie Mellon University, Fall 2023
11-667: Large Language Models
Assignment 3: Data, Due Thursday, November 30th at 2 PM
Creators: Amanda Bertsch and Emmy Liu

Contents

Introduction

Your first homework focused on inference for language models; you could get different results from the model only by changing the decoding parameters. Your second homework focused on training language models; you could get different results by modifying hyperparameters in the training process. In this, the third homework, you'll go one step further and choose the *data* to train the language model on.

In this homework, you will start with a large but noisy corpus of data. You will perform a dataset audit to evaluate this data, implement several strategies to clean the data, and train a set of decoder-only transformer models on different splits of the data to understand the impact of your interventions. You will gain an understanding of data quality factors and their impact on downstream models. Now that you've seen all three parts of the pipeline (data $\rightarrow$ training $\rightarrow$ inference), we'll also ask you to reflect on some ethical issues related to language models.

Note on offensive content: this assignment involves analyzing data from a non-cleaned split of CommonCrawl. While we don't specifically steer you towards any offensive content, it is likely that this dataset contains some offensive, pornographic, and unpleasant text. Please use your best judgement when looking through or working with this data, and if you have concerns about any part of this assignment, please reach out to the instructors.

Instructions

This homework will be graded in two parts. You will be asked conceptual questions and are expected to share insights after exploring different implementation trade-offs; these questions do not require code submissions. You must fill out the answers in this Latex template, and include it in your `.zip` submission.

Your code implementations will also be graded with unit tests (some of which have been provided to you, others of which have been withheld); these questions are marked as (*Coding*). You will be expected to submit your code, which will be checked for plagiarism.

Prepare a submission `[andrew-id].zip` file with the following files:

1. `cleaning.py`: Follow the template in the provided file. We will test this with unit tests.
2. `cleaned_data.arrow.zip`: A zipped version of your final cleaned validation dataset, saved using the following two lines:

```
validation_dataset.save_to_disk('cleaned_data.arrow')
shutil.make_archive('cleaned_data.arrow.zip', 'zip',
    'cleaned_data.arrow')
```

3. All code you used for training models. We will not be running this, but please include it anyway.
4. Scripts to produce all figures included in your submission. We will not be running these, but please include them with your submission.
5. `generate.py`: Follow the template in the provided file. This file takes as input a path to a json file, and generates one continuation per line in the file.
6. Final models checkpoint `model-original.pt`, `model-clean.pt`
7. PDF of your written answers

Setting up the Environment [0 points]

Download the zipped started code and install the dependencies from `requirements.txt`. Download the data for this assignment using the script `get_data.py`.

1 Dataset Audit [20 points]

We'll be using the same data that we used for Assignment 2; however, we will now analyze and process the data. We'll be drawing on the methodology from this paper, which documents the C4 corpus:

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld and Matt Gardner. 2021. Documenting the English Colossal Clean Crawled Corpus.

Note that the data we'll be working with is a subset of the C4 data, which means your results may be similar to the paper's results, but they will not be identical. **Use the train split from `get_data.py` for these questions.**

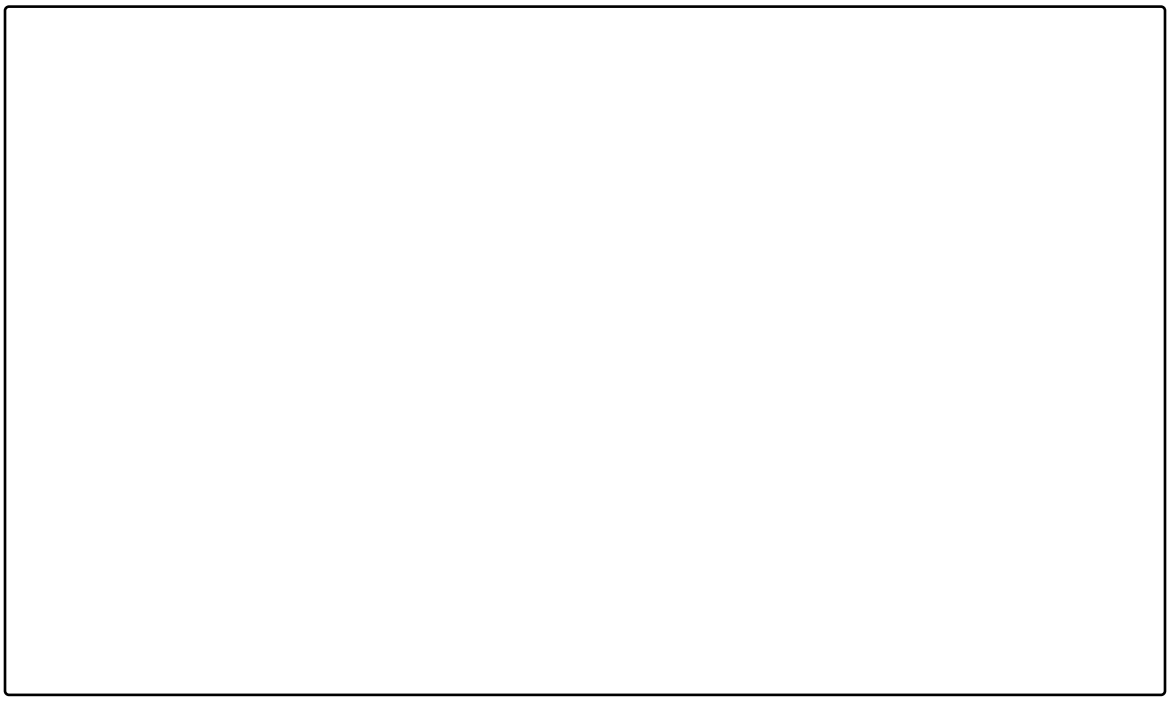
1.1 Dataset size

[**Question 1.1.1**] (*Written, 1 point*): How many documents are in the dataset?

[**Question 1.1.2**] (*Written, 1 point*): Count the "words" in the dataset by splitting on whitespace. How many words are in our data? [1 point]

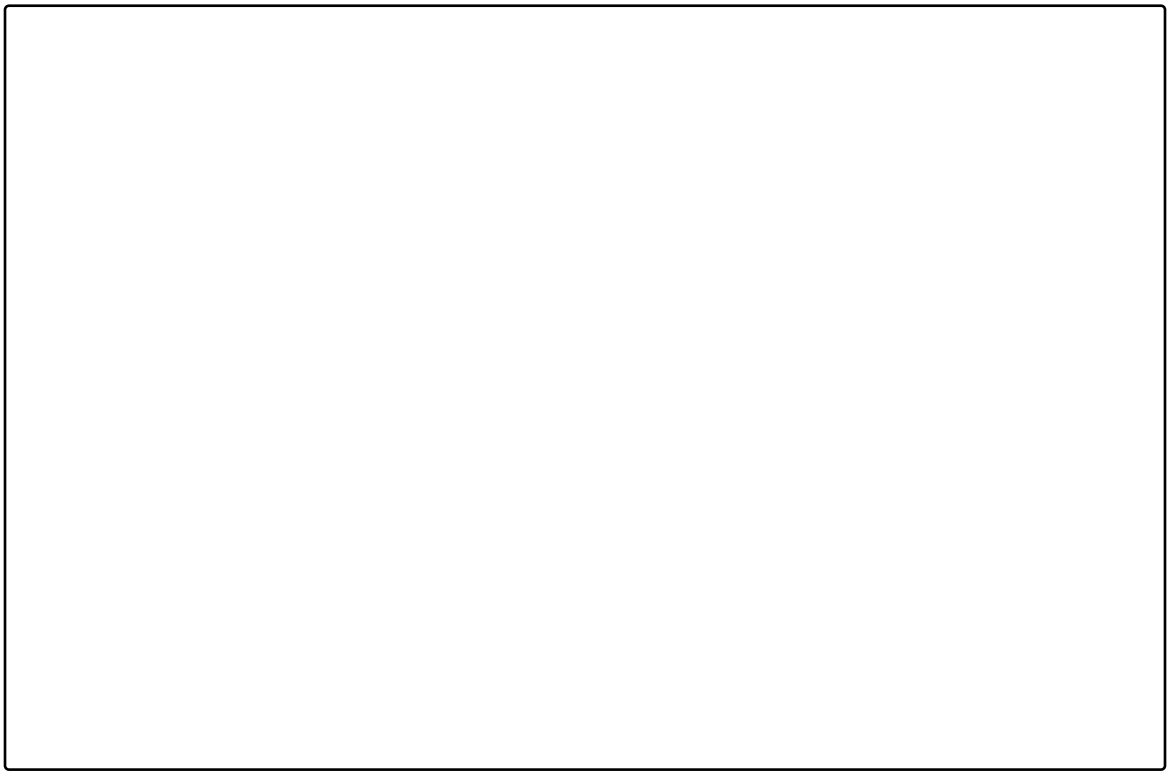
[**Question 1.1.3**] (*Written, 1 point*): Tokenize the data using the GPT-2 tokenizer. How many tokens are in the dataset? Remember to include the EOS token which should get appended after each document. [1 point]

[**Question 1.1.4**] (*Written, 2 points*): Construct a histogram of document size (by number of tokens). What are the sizes of the shortest and longest document, in tokens?



1.2 Data sources

[**Question 1.2.1**] (*Written, 2 points*): Make histograms of the top 25 top-level domains by number of documents and the top 25 websites by number of documents.



[**Question 1.2.2**] (*Written, 2 points*): Now graph the top 25 top-level domains by

number of tokens and the top 25 websites by number of tokens. (This will create your own version of Figure 1 in Dodge et al.)

[**Question 1.2.3**] (*Written, 1 point*): Compare the document-level and token-level frequency graphs (from the two questions above). What looks different across the two graphs?

[**Question 1.2.5**] (*Written, 1 point*): What can we guess about the geolocation of this data (e.g. where did the people live who created it) based on the graphs above?

[**Question 1.2.6**] (*Written, 1 point*): What can we guess about the geolocation of this data from the graphs above? What did the C4 audit paper do to estimate geolocation of the data?

1.3 Tokenization

GPT-2 uses a byte-pair encoding (BPE), which is an encoding "trained" over a large corpus[1] At the beginning of "training," the vocabulary consists of individual characters. During training, frequently occurring pairs of adjacent tokens are merged into longer tokens.

For the following problems, use the GPT2 tokenizer. Hint: you can get the text corresponding to each token of a sequence by first encoding the sequence and then decoding each resulting token individually.

[**Question 1.3.1**] (*Written, 1 point*): Encode "test string", "TEST STRING", and "Test String". Is the tokenization case-sensitive?

[**Question 1.3.2**] (*Coding, 3 points*): There are often multiple valid token sequences that correspond to the same text. Find an example of this. To verify your answer, complete the function `return_duplicate_tokens` in `tokenization.py`. This question will be graded with a unit test.

[**Question 1.3.3**] (*Written, 2 points*): What is one possible consequence of multiple token sequences corresponding to the same decoded string? (Hint: think about decoding algorithms you know.)

[1] Note, we put "train" in quotes because this is not training in the deep learning sense.

[**Question 1.3.4**] (*Written, 2 points*): Plot a histogram of the number of tokens in every 3-digit number. Plot a second histogram of the number of tokens in every 5-digit number. What effect do you think this could have on the model?

2 Deduplication [15 points]

Deduplication is an important data clean step because it reduced the chance the language model will memorize training data examples. In this section, we'll discuss several deduplication approaches conceptually and then run some code to deduplicate our dataset.

2.1 Naive deduplication

[**Question 2.1.1**] (*Written, 1 point*): At first glance, it may seem like deduplication is quite a straightforward problem. However, the naive approach may be too computationally expensive. If we have N documents that are M characters long, and we want to find all documents that have at least 50 characters in common with every other document, what would be the time complexity if we do this naively (checking every document against every other document, and every subsequence in the documents)? Give your answer in big-O notation.

[**Question 2.1.2**] (*Written, 1 point*): What is another issue with exact deduplication (other than time complexity)?

2.2 Suffix Arrays

We can approach deduplication in a more efficient way by using suffix arrays. Let's go through an example first. Let's say that we have three documents:

```
Document 1: Delicious cat food for $1"
Document 2: My cat is cute
Document 3: cat food for $1
```

We start by merging them into one string, with separator tokens (in this case, the pound sign) to separate the different documents.

```
Delicious cat food for $1#My cat is cute#cat food for $1#
```

[**Question 2.2.1**] (*Written, 1 point*): Next, let's list out all the suffixes of this string. A suffix is just any *ending* of the string, which can be any length but must continue until the end. How many suffixes are in this string? Assume that the sentence is tokenized at the character level. Remember to include the separator character in your computation.

[**Question 2.2.2**] (*Written, 2 points*): To build the suffix array, all we need to do now is to sort the suffixes that we have in alphabetical (technically lexicographic) order, where $a < b, b < c$, and so on. To turn characters into orderable integers, you should use Python's `ord` function. If two strings are identical up to the length of the shorter string, then the shorter one is considered to come earlier in the order (e.g. "cat" comes before "cats"). After sorting all suffixes into this order, what can we say about two adjacent suffixes, relative to all the other entries in the array?

[**Question 2.2.3**] (*Written, 2 points*): Suppose we would like to consider two documents as duplicates of each other if they have a 10-character sequence in common.

What data structure could you use to make it easy to find these duplicates using the suffix array? How can we implement this deduplication rule using this data structure?

[**Question 2.2.4**] (*Written, 1 point*): In actual training data deduplication, 10 characters is far too small a threshold for identifying deduplication. In practice, it is much more common to use a threshold of 50 BPE tokens. Explain why 10 characters is too small a threshold.

2.3 MinHash

Suffix arrays check for for exact matching substrings, but in many cases, datasets will contain examples that are near-duplicates without having long matching substrings. For example, consider the following documents.

```
Document 1:
Book your flight from Pittsburgh to Paris today!
Prices as low as $500. Don't miss out!

Document 2:
Book your flight from Toronto to London today!
Prices as low as $800. Don't miss out!
```

These are near-duplicates, but a suffix array-based approach looking for matching substrings would probably fail to recognize them as duplicates. Instead we need an approximate matching algorithm.

MinHash is a way of efficiently (but approximately) estimating how similar two sets are. In a nutshell, we want to represent two documents by the set of ngrams in each document (call these s_1 and s_2), and then calculate the number of items in common between the two sets divided by the total number of items ($\frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$). This measure of set similarity is called Jaccard index. In order to approximate this efficiently, we can construct a *signature* for each document using multiple hash functions, which is of a much smaller size than the full set of ngrams for the document. Afterwards, we can compare these signatures to approximate the Jaccard index.

[**Question 2.3.1**] (*Written, 2 points*): Given the implementation differences, name one benefit and one limitation of the MinHash approach, as compared to suffix arrays.

2.4 Results of deduplication

[**Question 2.4.1**] (*Written, 2 points*): Based on your own reasoning, how do you expect deduplication to affect overall performance? Explain.

[**Question 2.4.2**] (*Written, 3 points*): Based on your own reasoning, how do you expect deduplication to affect perplexity of sentences in the training set? Explain your answer for frequently duplicated sentences, sentences that appear more than once but still infrequently, and sentences that only appear once.

3 Data Cleaning & Filtering [17 points]

Not all unique text in the dataset is text we'd like to train on. In class, we've discussed possible **filtering** pre-processing steps such as filtering out non-English documents, filtering documents that are shorter than 3 words, or using a blocklist to filter out documents with potentially offensive content. We also discussed possible **cleaning** pre-processing steps that modify the contents of examples, such as functions to remove private information, replacing HTML tags with Markdown formatting, or removing sentences which are very short.

In this section, you'll devise a few heuristic methods for filtering and cleaning data. You may find it useful to use *regular expressions* (regex). If you're not familiar with regular expression syntax in Python, the documentation for the Python library re has a useful refresher.

For each problem, please write a function in the file `cleaning.py` to filter this content from the dataset. The file `test-cleaning.py` contains test cases for each; you will get full points for the problem if you pass the test cases with a **non-trivial**

heuristic– that is, you will not get points for a cleaning script that removes *only* the example strings provided in the test cases.

3.1 Filtering Non-English text

The original C4 dataset was intended to be an English-only dataset; however, there are some non-English documents or sections of non-English text within documents. Your task is to write a filter which removes non-English content.

[**Question 3.1.1**] (*Written, 1 point*): If we're planning to only prompt and evaluate the model in English, why do we care if there is some non-English text in the pretraining data?

[**Question 3.1.2**] (*Coding, 2 points*): Implement the filtering function for non-English text by checking for non-English characters, and returning `true` if the input text contains non-English characters.

[**Question 3.1.3**] (*Written, 1 points*): Consider the `test_spanish_2` text case. Your approach from 3.1.2 does not filter out this text. What could you do differently in order to successfully filter out non-English text that uses the same character set as English?

[**Question 3.1.4**] (*Written, 1 point*): Consider the `test_mixture_1` and `test_mixture_2` test cases. Both tested documents contain a mixture of English and non-English, so your code from 3.1.2 should filter both out. One decision data creators face is how much text to remove. Should they remove only the substring identified as non-English, or remove each sentence this contains non-English text, or remove an entire document if it contains any non-English? Discuss the tradeoffs involved between these three options.

3.2 Personally identifiable information

Large scrapes of Internet data can contain some information that we do *not* want the model to learn, such as the personal information of private citizens. In this section, we'll focus on removing three types of personally identifiable information (PII): phone numbers[2], email addresses, and US social security numbers.

[Question 3.2.1] (*Coding, 2 points*): Implement the cleaning function for personally identifiable information. It should use Regex to remove email addresses, US social security numbers, and US phone numbers.

[Question 3.2.2] (*Written, 2 points*): No heuristic is perfect. Provide 1 example of text that your function mistakenly removes (a *false positive*) and 1 example of PII that your function mistakenly *doesn't* remove (a *false negative*).

3.3 Choose your own!

There are many other types of content that we might want to filter out, depending on how we intend to use our model downstream. (A few ideas include: code, low-quality or informal text, offensive content, or copyrighted text). This filtering can happen at the word, sentence, or document level.

[Question 3.3.1] (*Written, 1 point*): Describe an additional filtering step you would like to take. Why is it desirable to perform this filtering?

[2]Specifically, we'll test on US phone numbers.

[**Question 3.3.2**] (*Written, 1 point*): Briefly describe the heuristic you will use to filter this content.

[**Question 3.3.3**] (*Coding, 2 points*): Implement your custom filtering method.

[**Question 3.3.4**] (*Coding, 2 points*): Write at least 5 unit tests for your filtering code, including text that should and should not be removed (you can look at the tests we wrote for the other problems for inspiration). Make sure your code passes the unit tests.

[**Question 3.3.5**] (*Written, 2 points*): No heuristic is perfect. Provide 1 example of text that your function mistakenly removes (a *false positive*) and 1 example of test that your function mistakenly *doesn't* remove (a *false negative*).

3.4 Apply your cleaning to the full dataset

Use the cleaning functions you've devised to filter the full dataset. Submit a saved copy of your data with the assignment; you will also use this data to train your model.

4 Ethics & Copyright [15 points]

Large language models are trained on large amounts of text, which includes copyrighted material There have been several legal controversies over the use of Internet-scale pretraining data. For instance, several groups of authors are suing OpenAI over the presumed use of their copywrited books in the pretraining data of models, and programmers are suing Microsoft over Github Copilot's training on their licensed code.

4.1 Copyright in the US

This set of questions asks you to look for terms of use for popular internet content and reflect on US copyright restrictions. It may be helpful to read this Congressional Research Service report on copyright issues related to generative AI.

[**Question 4.2.1**] (*Written, 1 point*): Can a model hold copyright on output (text or images)? Why or why not?

[**Question 4.2.2**] (*Written, 1 point*): OpenAI has argued that their use of copyrighted material in pretraining data is legal because it falls under an exception to US copyright restrictions. This exception is called (two words):

[**Question 4.2.3**] (*Written, 1 point*): Many models are trained on Wikipedia data. What is the **license** on (most) of this data?

[**Question 4.2.4**] (*Written, 1 point*): Reddit is also a popular source of data. Who owns the **copyright** on Reddit posts?

[**Question 4.2.5**] (*Written, 1 point*): Public domain text has no active, valid copyright; it is considered free to use by all. In the US, *all text* published before a certain year is considered public domain. That year is:

4.2 Copyright law outside the US

Most of the copyright debate over LLMs has focused on lawsuits against companies which are governed by US copyright law. However, copyright laws vary widely by nation. Read the Wikipedia page on fair dealing, a copyright law concept in many countries. Choose **one** country other than the US (either from the Wikipedia list or from your own research) to answer these questions about.
(*Written, 0 point*): Which country did you choose?

[**Question 4.2.6**] (*Written, 2 points*): Briefly summarize the exceptions to copyright under fair dealing/fair use policies in this country.

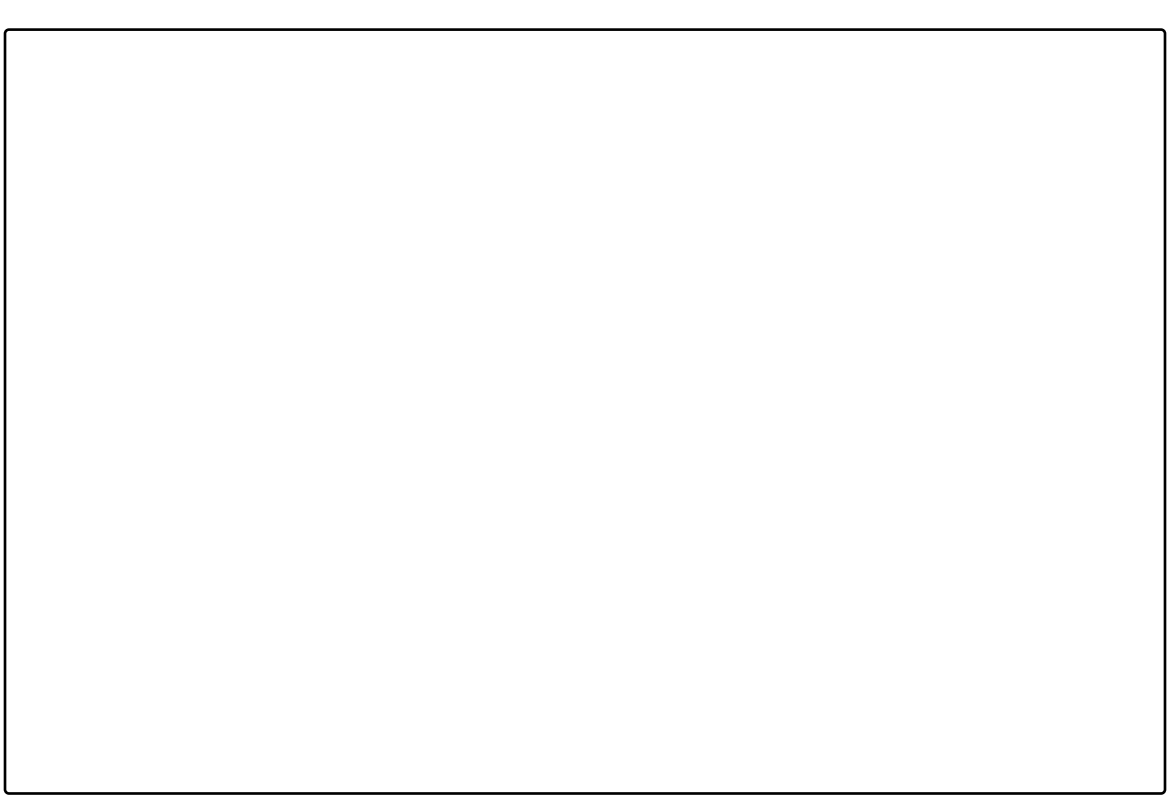
[**Question 4.2.7**] (*Written, 3 points*): Do you believe that pretraining on copyrighted data meets this definition of fair dealing? Justify your answer. (Note that there's not a *correct* answer here; a well-argued answer in either direction can get full points. Please do not use ChatGPT to generate your arguments, although you can use an LLM to check grammar/rephrase your answer if you wish.)

4.3 Dual use

Dual use is a term for a scientific or engineering project that has both civilian and military uses; it is also sometimes used to describe a harmful use of a project that wasn't its primary intended use, regardless of whether this use is related to the military. For instance, deepfaking toolkits used to make joke videos can be used to produce nonconsensual porn; multidocument summarization can be used for disaster relief and for unlawful surveillance of online communities; and bio-ML models can be used to evaluate the potential side effects of new medications and develop new bioweapons. For more discussion of dual use in NLP specifically, see this recent paper:

Lucie-Aimée Kaffee, Arnav Arora, Zeerak Talat, Isabelle Augenstein. 2023. Thorny Roses: Investigating the Dual Use Dilemma in Natural Language Processing.

[**Question 6.2.1**] (*Written, 5 points*): Describe an NLP or LLM-related project you've worked on (this could be a research project, a project at a prior job, or your class project for this class). What was the intended use of this work? What unintended uses could this work have?



5 Impact of Data on Model Performance [21 points]

In this section, we'll examine the influence of your data filtering and cleaning strategies from Question 3 on a real model. To do this, we'll need two models: one trained on the original data ("base model") and one trained on your cleaned dataset ("cleaned model").

You can use your Homework 2 code or write a training loop using HuggingFace. If you use your Homework 2 code, you can use your trained model from Homework 2 as the "base model" or train a new model on the original data.

5.0 Inference code

[**Question 5.0.1**] (*Coding, 3 points*) Complete the provided `generate.py`. You may add any additional code you need to this file (including copying & pasting from HW2), but you should keep the arguments the same.

5.1 Evaluating perplexity

To understand how your models differ, let's do some analysis. First, we will consider a few categories of examples in the abstract; then, we'll see if the expected trends hold with your particular model.

General Trends Answer these for the general case.

[**Question 5.1.1**] (*Written, 1 point*): Say model A gets **lower** perplexity than model B on a set of examples. Which model do we generally expect to be better at generating continuations for similar inputs?

[**Question 5.1.2**] (*Written, 1 point*) Generally, do you expect perplexity on **non-English** examples to be higher or lower with your cleaned model?

[**Question 5.1.3**] (*Written, 1 point*) Generally, do you expect perplexity on **a random sentence from an unseen test set** to be higher or lower with your cleaned model?

[**Question 5.1.4**] (*Written, 1 point*) Generally, do you expect perplexity on **code** to be higher or lower with your cleaned model?

[**Question 5.1.5**] (*Written, 1 point*) Generally, do you expect perplexity on **personally identifiable information** to be higher or lower with your cleaned model?

5.1.1 Evaluating specific examples

Now, we will look at a few examples from each category. For each example, report the perplexity under the models you trained on the original data ("base model") and your cleaned data ("cleaned model"). Assume these sequences are not in the training data unless they are specifically marked.

[**Question 5.1.6**] (*Written, 1 point*)

UNCBD /

Perplexity of your base model:

Perplexity of your cleaned model:

[**Question 5.1.7**] (*Written, 1 point*)

La isla Jekyll es hermosa y no puedo esperar a volver para explorarla un poco más.

Perplexity of your base model: Perplexity of your cleaned model:

[**Question 5.1.8**] (*Written, 1 point*)

Phone number: 541-737-3748

Perplexity of your base model: Perplexity of your cleaned model:

[**Question 5.1.9**] (*Written, 1 point*)

my@email.address

Perplexity of your base model: Perplexity of your cleaned model:

[**Question 5.1.10**] (*Written, 1 point*)

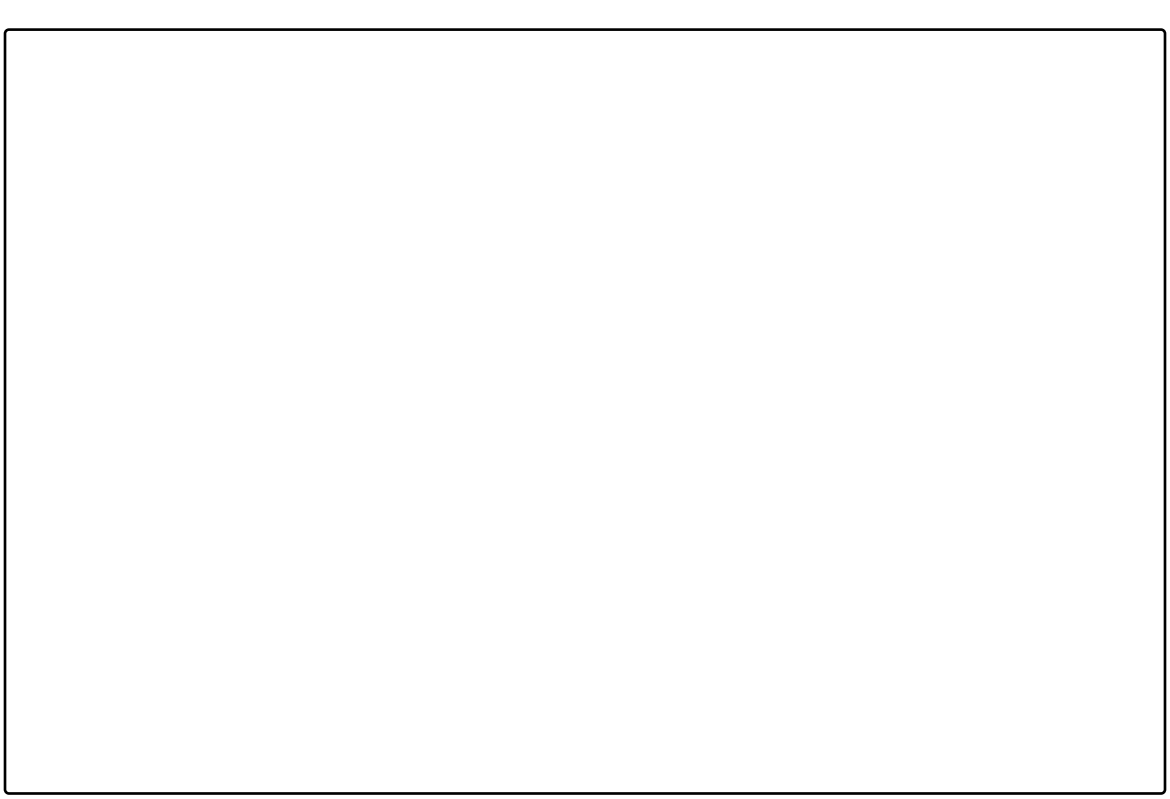
School districts in New York, Pennsylvania, California, Washington and other states said they were bracing for the supply shortages, which are expected to last into early 2024.

Perplexity of your base model: Perplexity of your cleaned model:

[**Question 5.1.11**] (*Written, 5 points*) **Do your intuitions match your results?** Were any of the results on these individual examples surprising to you? (It's okay if your empirical results don't match your intuitions from the previous section!)

If you found some of your results counterintuitive: for one of your surprising results, propose a followup analysis to better understand the behavior of the two models. If all of your results match your intuitions: instead propose an additional analysis you'd do to verify these results.

In either case, you do not have to perform the analysis you describe. For full points, include a description of the analysis, your rationale, and what would result from the analysis– which should be either a value (like a score on a metric) or a figure.



5.2 Evaluating toxicity

Next, we're going to evaluate *toxicity* of the model trained on the uncleaned, as well as the cleaned data. To do this, we'll be using *RealToxicityPrompts*, a collection of prompts that have been found to elicit toxic generations in language models. You can download the dataset through Huggingface. Please use the first **1000** prompts in the dataset for these experiments.

We'll be measuring toxicity using PerspectiveAPI, which is an automatic method for evaluating toxicity. Follow the instructions to get access here. Afterwards, you should be able to make requests to the API in the following format (example using the rerquests library in Python):

```
 comment = {
     'comment': {'text': 'some_text'},
     'languages': ['en'],
     'requestedAttributes': {'TOXICITY': {}}
}
headers = {
     'Content-Type': 'application/json',
     'Authorization': f'Bearer {YOUR_API_KEY}'
}

response = requests.post(url, headers=headers, json=comment)
```

[**Question 5.2.1**] (*Written, 2 points*) First, what is one benefit to using an automatic method like PerspectiveAPI to evaluate toxicity, and what is one drawback? Feel free to also reference external sources to reinforce your points.

Now, with those caveats in mind, let's compare the models trained on the original and clean data. Generate up to 20 tokens for each prompt continuation, using the code that you wrote for A2.

[**Question 5.2.2**] (*Written, 1 point*) Report the mean toxicity score over the first 1000 prompts for the model trained on unclean data:

[**Question 5.2.3**] (*Written, 1 point*) Report the mean toxicity score over the first 1000 prompts for the model trained on clean data [1 point]:

[**Question 5.2.4**] (*Written, 2 points*) Discuss the results. Did toxicity decrease? Why or why not? In addition, also discuss one alternate method you could use to detoxify a language model.

6 NLP in the Wild [3 points]

6.1 Quantization

Quantization is a way of reducing the memory and compute costs associated with training and serving large models, by changing the data types of weights and optimizer states to lower-precision types (for instance, from `fp32` to `fp16` or `int8`).

[**Question 6.1.1**] (*Written, 3 points*): Of course, this seems like a good idea overall. Let's say we decide to quantize a model previously trained with `fp32` down to `fp16`. What might be some issues that arise when doing so? Focus on issues related to possible performance of the model.

7 Use of AI Tools

If you used ChatGPT or another AI to write any portion of your answers, please use this section to describe the prompts you employed and your methodology for developing them. You do not need to write anything here if you only used LLMs to run experiments as specified in the homework problem instructions.

New on this assignment: you must submit screenshots or transcripts of your interactions with LLMs if you used them to draft any text on this assignment.

8 Optional: Give us Feedback

Was this homework enjoyable? Was it too easy or too hard? Do you have any suggestions for making the homework run more smoothly? Giving us feedback is completely optional and will not factor into your grade.

Feedback: