Intro

Attention patterns are somewhat unique in transformers in that they are mappable back to individual tokens [n_ctx n_ctx]. This makes them amenable to visual interpretation. However, these patterns are part of a larger computation, contributing to an internal representation that the model uses to generate the output and we can't take this too literally.

Nonetheless, visualizing and generating data on attention patterns can be beneficial for understanding and interpreting the model's behavior. The `AttentionData` class is useful if you want to:

- **Automatically generate and pass a prompt with relevant stats to a GPT for a first hypothesis at the head's behavior**
- **See which tokens an attention head activates the most or least on (weighted by current sequence length)**
- **Quickly visualize 1D (instead of 2D) attention patterns and in different contexts**
- **See the distribution of how much a head scores a particular token and in what contexts**

Importantly, the core AttentionData class can be used with any arbitrary combination of dataset (provided it is List[List[str]]), `HookedTransformer` instance, and OpenAI GPT model. Please note this is a hackathon project and thus may contain bugs and is unlikely to be maintained or developed further.

***Note: The unreadable black cells in the below output are just a result of the pdf conversion, they are viewable at the demo notebook: https://github.com/connor-henderson/attention-data/blob/main/demo.ipynb**

Setup

In [1]:

```
from attention_data import AttentionData
import os
import openai
import torch as t
from transformer_lens import HookedTransformer
%pip install python-dotenv
from dotenv import load_dotenv

load_dotenv()

# Set API Keys
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY", "")
assert OPENAI_API_KEY, "OPENAI_API_KEY environment variable is missing from .env"
openai.api_key = OPENAI_API_KEY

# Saves computation time, since we don't need it for the contents of this notebook
t.set_grad_enabled(False)

device = t.device("cuda" if t.cuda.is_available() else "cpu")
```

```
Requirement already satisfied: python-dotenv in ./env/lib/python3.10/site-packages (1.0.0
)

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [2]:

```
# Get a dataset

%pip install datasets > /dev/null
from datasets import load_dataset
dataset = load_dataset("stas/openwebtext-10k", split="train", trust_remote_code=True)
```

[notice] A new release of pip is available: 23.2.1 -> 23.3.2

```
[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

```
# Get a model

model = HookedTransformer.from_pretrained("gpt2-small")
```

```
Loaded pretrained model gpt2-small into HookedTransformer
```

Usage

(Note: currently the first token attention seems overly high, might be a bug)

At A High Level

Instantiate an `AttentionData` class with your chosen passed parameters and call any of the following:

- `describe_head`
  - **params:** `head=0, layer=0, num_samples=10, custom_prompt=None, print_description=True`
  - **Creates a prompt based of `num_samples` of examples and returns (prompt, description) where the description is the GPT's guess at the themes of the attention patterns**
- `get_ranked_multiples`
  - **params:** `layer=0, head=0, str_token=None, num_multiples=10, reverse=False, display=False`
  - **Returns and displays the top (or bottom if `reverse=True`) `num_multiples` number of tokens with the highest score multiples, optionally pass in a specific str_token to only return instances of that token**
- `get_random_multiples`
  - **params:** `layer=0, head=0, num_multiples=10, display=False`
  - **Returns and displays a random `num_multiples` number of tokens**

In [5]:

```
# Make an AttentionData instance

attention_data = AttentionData(
    model=model,
    text_batch=dataset['text'][:100], # Speed is sensitive to the number of samples
    openai_model="gpt-3.5-turbo-1106",
    openai_api_key=OPENAI_API_KEY,
    suppress_first_token=True, # Temporary hack for denoising, unprincipled
)
```

In [ ]:

```
# Let's look at L10H7, which was studied closely here: https://arxiv.org/pdf/2310.04625.pdf
layer = 10
head = 7
```

In [6]:

```
# Any first method call on attention_data will be slowest since has to generate the cache

prompt, description = attention_data.describe_head(layer=layer, head=head, num_samples=20)
```

```
Creating new samples for layer 10 head 7
```

```
Token indices sequence length is longer than the specified maximum sequence length for this model (5989 > 1024). Running this sequence through the model will result in indexing errors
```

```
Making API call to gpt-3.5-turbo-1106...

Based on the attention patterns observed, it appears that the attention head in
the transformer focuses on several key aspects. Firstly, it pays attention to to
kens that represent proper nouns and entities, such as names of individuals and
organizations, which are crucial for understanding context and relationships. Ad
ditionally, there is attention towards tokens that signify actions or significan
t events in the text, indicating a focus on verbs and action-related words. The
attention scores also suggest an emphasis on the beginning and end of sentences,
 potentially capturing the importance of sentence boundaries and transitions. Ov
erall, the attention head seems to prioritize information-carrying tokens, espec
ially those relating to entities, actions, and sentence structure, to effectivel
y process and generate coherent text.
```

In [14]:

```
# The "multiple" is the multiple of the average attention pattern value for a row,
# i.e. a multiple of 2 in a row with 10 tokens means the attention score was 0.2

ranked_multiples = attention_data.get_ranked_multiples(
    head=head,
    layer=layer,
    num_multiples=10,
    display=True
)
```

Layer 10 Head 7, Top 10 / 17031 Multiples

Token	Multiple of Avg. score	Pattern
in	29.0	
measure	29.0	
by	29.0	
now	29.0	
their	29.0	
children	29.0	
B	29.0	
on	29.0	
week	29.0	
3	29.0	

In [16]:

```
# Look at the top occurences for a particular token
example_str_token = ranked_multiples[4][0]

ranked_multiples = attention_data.get_ranked_multiples(
    head=head,
    layer=layer,
    num_multiples=10,
    str_token=example_str_token,
    display=True
)
```

Layer 10 Head 7, Top 10 / 17031 Multiples for ' their'

Token	Multiple of Avg. score	Pattern
their	29.0	
their	13.6	

their	12.7
their	8.8
their	8.4
their	7.7
their	7.1
their	6.7
their	6.4
their	6.3

In [18]:

```
# Look at a random grouping of multiples that were larger than average

random_multiples = attention_data.get_random_multiples(
    head=head,
    layer=layer,
    num_multiples=15,
    display=True
)
```

Layer 10 Head 7, 15 / 17031 Random Multiples

Token	Multiple of Avg. score	Pattern
this	2.6	
than	2.0	
about	3.7	
Gold	3.1	
in	1.6	
Wednesday	3.9	
PL	2.1	
vessel	2.8	
Dept	11.5	
humanity	5.4	
up	3.0	
really	1.2	
of	1.5	
Wednesday	2.6	
	4.1