EleutherAI

Onboarding

Hey there! Welcome to EleutherAI (A.K.A `OpenAI: Oops All Open!`) We're trying, among other things, to build **huge open source Language Models** (i.e like GPT-3, and possibly even bigger), if you have any questions check out the various channels on our discord or just ask. And if you want to contribute we're always grateful! Below is a short description of our goals, and the tasks that need doing. The github for the model itself is currently private but if you have experience working with Tensorflow on TPUs (or tensorflow-mesh specifically.), we could use all the help we can get.

Current projects

Our current projects are:

- A **GPT-3 Replica** (naming: to be determined): this is the main flagship project for now.
- **Other large LMs** (1B-100B): Before we replicate GPT3, we will be training a series of smaller models, ranging from GPT2 size up to a significant fraction of GPT3 size. We tensorflow-meshmight also try out other model architectures.
- **The Pile™**: A balanced dataset of many different sources for Language Modeling. The goal is to provide exposure to as diverse a range of text modalities as possible.
- **HUMONGOUS** (**H**umongous lang**U**age **MO**delling inter**N**et **G**eneral purp**O**se **U**se data**S**et): A dataset of web-based text extracted from CC and ready for Language Modelling. Will be filtered to create a major component of The Pile™.

Warning: parts of this document may be outdated since this project moves fast.

TOC

The Current Priorities

(subject to change, most likely really often! This project moves fast.)
(if you see something out of date, please update it!)

- GPT3: we need to figure out why our GPT2 models aren't reaching the loss they're supposed to, despite supposedly having the same data. Lucidrains has replicated the issue with newly encoded data so it doesn't seem like a data/tokenization problem. Currently working on sampling to rule out the chance of loss just being misreported.
- GPT3: our efficiency is below what Colin Raffel says mtf should be able to get. Could be a layout issue, or it could be something else we're doing wrong. Worth combing through the mtf transformer code to figure out what we're doing differently
- HUMONGOUS: html-to-text benchmark needs more data, especially for low resource languages!
- Books2: we need to decide on a pdf-to-text converter pipeline. Quite a few options, we already have *something* working but it may or may not be the best/fastest.
- Books2: Also, we need to figure out how to get the data from libgen onto a machine we control in a reasonable timeframe and without inviting legal trouble. Maybe we can contact some people from r/DataHoarders?)

New onboarding doc:
https://docs.google.com/document/d/1yOnxEMlU57M8YFlQC3XNOvyMVX2EpU5LeIWhEBcwQNk/

The original brainstorming document:

This page was an early brainstorming document. Parts of it may be <u>no longer relevant</u> but it shows our thought process and contextualizes the project goals.

What we ended up deciding on: see HUMONGOUS and The Pile™ sections for data info; see TF mesh section for TPU training info

- Compute: (ESTIMATE IS FOR 1T) by OpenAI estimates, we'd need 3.64E+03 * 1/0.175 = 20800 PF-days of compute; with a v3-2048, assuming perfect utilization, that's 20800 / 107.52 = **192 days**
- To be honest, **funding**. If you're interested in the project and want to help us along, server costs are going to be high, as we're storing & processing a lot of data. (TODO: set up patreon / ko-fi.) We also need **developer time**. If you are a developer, your help would be appreciated. The main thing that needs help is mesh tensorflow; pop by #gpt-neo for more info on that. Help with data stuff (pdf cleaning, etc) would also be appreciated. You can ask in #the-pile for more info.
- Things we need funding for:
    - Google cloud buckets. The final data will be ~6TB over about one year—more if we can only get our hands on smaller TPU pods and can't set up swarm training or something. This is going to be **expensive**.
    - CPU cores: processing CC and Books2™ data and tokenizing to tfrecords will eat a LOT of CPU power. We're renting from hetzner which already cuts our costs vs GCP, but it's still going to cost a lot.
- Data:
    - Data gathering, see Data Gathering, Data Cleaning & Data format sections.
    - Decision: do we want to use BPE or codepoint level? We're going to try both on a smaller model
- Training
    - Model shape & training hyperparameters (consult OA literature about model scaling, shouldn't be too hard)
    - Decision: which efficient transformer techniques do we use? I.e Reformer, linear attention, OA sparse attention, routing transformer, etc.
    - How do we implement that? (might be a lot of work, depending on which ones we pick!)
    - Decision: which training tricks do we use? I.e partitioning individual layers across TPUs, GPipe, L2L, grad checkpointing, swarm stuff, etc. (the answer to this depends on what shape the model is, whether individual layers fit on TPUs, etc)
    - How do we implement that? (certainly will be a lot of work, TPUs are annoying!)

Data Gathering for The Pile™

Training a big model means big data. We're looking to gather as much high-quality text data as humanly possible. If you can help us out with gathering any of the following datasets, or have any other suggestions for dataset sources we haven't already covered, please let us know.

See #datasources for sources of data we plan to scrape, and #data-scripts for already implemented scripts for data gathering.

Massive datasets (>100GB compressed):

- (WIP - currently downloading) Common Crawl: webpages; filtered as described in the GPT3 paper
- (WIP - downloaded to hetzner; someone needs to process) CORE: academic papers
- (WIP) Libgen (epubs only): books
- (WIP) Libgen (pdfs): books (see pdf cleaning section)

Large datasets (>10GB compressed):

- WebText (2?): high quality webpages scraped from reddit links
- Pushshift reddit comments
- (DONE) BookCorpus: books
- (DONE) Wikipedia (en)

Small datasets:

- IRC: chat logs; combination of many sources
- (DONE) Opencaptions

Unknown Sizes:

- (WIP) Youtube CC dataset
- Scrape links from twitter data:

How much data do we need

Here is an approximate estimate of how much data we need, based on guidelines from the OpenAI scaling paper. **Important: since we're not necessarily training models the same way as OpenAI does, their estimates are only useful as a rule of thumb.** We might need more or less data than this.

Params	Data (compressed size)
175B (GPT3)	400GB
1000B (1T LM)	1500GB = 1.5TB
10000B (10T LM)	9000GB = 9TB
100000B (100T LM)	54000GB = 54TB
1000000B (1Q LM)	324000GB = 324TB

Data Cleaning

To get a healthy model, we need clean data. Tasks to work on include:

- Dataset filtering (need to implement logistic classifier on WebText/CC to filter for high quality & book cover / content / index page filter.)
- Implement pipeline and integrate deduplication (using something like https://github.com/ekzhu/datasketch). For reference, this is what OA used: https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.feature.HashingTF

Data Format for The Pile™

The Pile™ is the name of the dataset we plan together, consisting of a mishmash of different sources and hopefully resulting in a big ol' pile of high quality text data. Specs for The Pile™ are as follows:

- 1 Document = 1 Logically connected thing. Examples: IRC logs should be 1 channel = 1 document. Since irc logs are usually already split by date, you need to re-join them back together since that split is artificial.
- Data format: Make a json object where each element is a document. Compress that json using zstd (preferred), or gz. We plan to use this library to standardize all the data before training: https://github.com/leogao2/lm_dataformat
- Try to get data that probably wouldn't be in Common Crawl (which is essentially a random sample of webpages), though because we're downloading only a subset of Common Crawl, and we're deduplicating eventually, overlap isn't a big deal.

- Please package your code up into a repository where as little human intervention as possible is necessary. Automate everything so you can generate the final data with a single command. See #datascripts in the discord for example scripts.
- Rule of thumb for size: Anything from 100MB to 100GB is useful, though bigger data is more useful.

MODEL:

Arch & Tricks:

Local attention

- Local window, maybe dilated: could work, maybe difficult to implement *efficiently* for TPU. will probably need another type of attention too to intersperse. Maybe use convolutions to optimise?

Global attention layers

- Reformer: could work, might be slow, maybe difficult to implement for TPU
- Routing Transformer: Only a modest complexity savings
- Sinkhorn Transformer: ?
- Linear attention: Easy to implement, Not useful because our d_model is bigger than context length
- Compressive transformer: might be especially useful for char-level where information of each character is low

Parameters that don't get used on every step

- MoE: could work, maybe difficult to implement for TPU (unless google releases Gshard)
- Product Key Memory: performance? ; official code is available, but only pytorch

Fitting stuff into memory

- L2L: might be useful, but predicated on where we store the weights and how fast we can get them to/from the TPUs
- GPipe: optimizes long chains of dependent steps (i.e layers), implementation exists
- GShard: could enable *much bigger than* 1T size, but might be very difficult to implement (modifies XLA)
- TF Mesh: probably easiest to get working. Splits operations across devices (?)

GPT-MESH:

We're attempting to build a large model with the tensorflow-mesh library (https://github.com/tensorflow/mesh). It's very badly documented, we're slightly out of our depth, and there's like three publicly available models total built with the library. We would love help from anyone who thinks they can contribute. Message @Daj if interested.

TODO:

- Sampling

CPU offerings:

The below users have offered us CPU power for data processing when the time comes @ceeps @zitterbewegung @Ruby @zphang (Computer at University?) @jili @shawwn (server) @GalaxyBrainHuman (too many computers at home) @aquajet

HUMONGOUS: **H**umongous lang**U**age **MO**delling inter**N**et **G**eneral purp**O**se **U**se data**S**et

What is this?

- Convert all of CC to usable high quality text.

But doesn't CC already provide WET files which are text only?

- Yes, and they are *horrendous* in quality. Basically unusable for LM training. Also, filtering text files is really hard.

What about CC_net or OSCAR or C4 or etc?

- Those all process only a single snapshot of CC or some other small sample. We want to eventually process all of it, to provide a base dataset for future LM training.
- Also, these datasets often do some *really* aggressive processing. Our goal is to eliminate *most* of the cruft so researchers can filter, however they want to, a 35TB dataset (number only for illustration purposes), rather than a 3500TB one.

How will we deal with non-English languages? Filter or keep?

- Keep; we're currently developing a new pipeline to handle multilingual, because justext chokes on lots of languages. Also, newspaper sucks at forums, etc. We're currently working on a benchmarker to tell us how good each extractor is at each type of site and language, and using that info to maybe tune/combine/develop extractors.

How big is all of CC in WARC form?

- About 3.5PB

How big will the resulting text be?

- We don't know for sure yet, but probably a few dozen TB.

How does this tie into The Pile™? Is this the same thing?

- The CC slice of The Pile™ will come from CCTC. The rest from elsewhere.

But it will take ages!

- Yes it will, but we don't need to do it all at once. I have split the dataset into chunks and once we collect enough chunks for The Pile™ we can start on that. Then we can keep collecting chunks until we complete CCTC.

PDF Cleaning

- Headers and footers: we dont want em. This includes page numbers, etc. Footnotes are a bit of an edge case and I don't have any strong opinions either way
- ocr: lots of these pdfs are scans and they already have ocr but it's absolute garbage. if we can find a really good ocr it might be worth it to do our own ocr. Maybe being able to detect whether the embedded ocr or custom ocr is better would be nice too
- math formulas! most pdf-to-texters absolutely mangle them. if there's some ocr to turn math into latex or something that would be *amazing*.
- tables: at the very minimum we want to be able to detect tables and replace with "[insert table here]" instead of vomiting numbers over the page.
- detect probably mangled text and remove it
- handle multicolumn text
- fix formatting issues like hyphenation, random spaces, etc. one issue with hyphenation: sometimes letters get duplicated, sometimes they don't. I.e "hyph- enation" vs "hyph- henation"
- Moonshot idea: train a network to detect graphs. train another network to summarize the important content of graphs in plain english (given as input: surrounding text for context, any ocred text in the graph itself, and an image of the graph itself). Example: "Figure 1: Train and test error as a function of model size, for ResNet18s of varying width on CIFAR-10 with 15% label noise. The diagram depicts a solid blue curve, indicating the test performance, getting worse briefly before continuing to drop. A dotted line for training error goes to zero. A red band labelled critical regime..." use that to replace all graphs with summaries. (also, something similar for tables would be great too)

Potential projects we can do:

- Impacts of context length on performance
- A better html->text extractor + benchmark suite (https://github.com/leogao2/htmltotext-benchmark) - I'm thinking we can make an ensemble extractor using a classifier to decide which to use, or something.
- A better pdf->text extractor + benchmark suite?
- A post-hoc text cleaner to get rid of mangled text, boilerplate, etc from text already extracted from html or pdf
- Rad Lab stuff
- Model architecture stuff (local attn, all-attn memory key-values, axial positional embedding, moe, GLUs, ReZero)
- GPTuring (Training or augmenting a LM on program outputs)

Notes from video (https://youtu.be/mE8UK7bukME):

- 2-3x improvement per flop over gpt2 for same size model (I'm guessing probably due to sparsity, he seemed to suggest that data might have been a factor too.)
- Books1 and Books2: mostly fiction
- Books1: english; Books2: "pretty much anything"
- OA plans to do multilingual in the future
- Even when filtered, CC still "has problems" ("webspam leak through")
- Using only CC hurts downstream performance a lot
- Mixing in a tiny fraction of another dataset gets most of the benefit in terms of skills
- What was the bug? The test set decontaminator would only remove one instance of any matched contamination and miss all the other ones
- Perf/flop still low for gpt3
- Poor multilingual performance - OA also didnt have time to collect high quality multilingual data. OA also wants to do multilingual in the future for reasons similar to us
- GPT3 does poorly on some tasks, such as entailment; hypothesis: denoising objective (MLM) creates "richer" representation
- Didnt publish finetuning because: did run experiments, generally "does ok", can be really valuable for application, larger models can still do things that smaller models cant do after fine tuning but results arent as dramatic (?)
- Common sense - not much physical reasoning on the internet - more a problem of the training data
- Dataset quality - could use a lot more work. Logistic regression between CC and WT not optimal, could use more work
- 42TB of CC data as "possible input"- the amount left after extracting all of CC 42TB -> 600GB
- OA has had 12B for over a year now
- Physical reasoning: "lots of ideas, cant talk about them" - he mentions that people get this information through experience irl - my bet is that OA is probably working on multimodal, but that might not be done soon; in the meantime they might try to (and we should try to!) construct a physical reasoning dataset
- Big models are better at *storing world knowledge*
- OA is keeping info about training parallelism vague to "keep everything safe" by preventing other