03:10:21 datasets used in GPT-2, GPT-3, FineWeb (EDU)

take a look at what data sets were used by gpt2 and gpt3 so gbt2 used this web Text data set that was never released um there's an attempt at reproducing it called open web text uh so basically roughly speaking what they say here in the paper is that they scraped all outbound links from Reddit and then uh with at least three Karma and that was kind of like their starting point and they collected all the web P all the web pages and all the text in them and so this was 45 million links and this ended up being 40 GB of text so uh so that's roughly what gpt2 says about its data set so it's basically outbound links from Reddit now when we go over to gpt3 there's a training data set section and that's where they start to talk about um common coll which is a lot more uh used actually I think even gpt2 talked about common coll um but basically it's not

a very high quality data set all by itself because it is extremely noisy this is a completely random subset of the internet and it's much worse than you think so people go into Great Lengths to filter common craw because there's good stuff in it but most of it is just like ad spam random tables and numbers and stock tickers and uh it's just total mess so that's why people like to train on these data mixtures that they curate and uh are careful with so a large chunk of these data mixtures typically will be common C like for example 50% of the tokens will be comic but then here in gpt3 they're also using web text to from before so that's Reddit outbound but they're also adding for example books and they're adding Wikipedia there's many other things you can decide to add now this data set for gpt3 was also never released so today some of the data sets that I'm familiar with that are quite good and would be

representative of something along these lines are number one the red pajama data set or more specifically for example the slim pajama subset of the red pajama data set which is a cleaned and D duplicated version of it and just to give you a sense again it's a bunch of common crawl um C4 which is also as far as I know more common craw but processed differently and then we have GitHub books archive Wikipedia stack exchange these are the kinds of data sets that would go into these data mixtures now specifically the one that I like that came out recently is called Fine web data set uh so this is an attempt to basically collect really high quality common coll data and filter it in this case to 15 trillion tokens and then in addition to that more recently huggingface released this fine web edu subset which is 1.3 trillion of educational and 5.4 trillion of high educational content so basically they're trying to filter common C to very high quality

educational subsets and uh this is the one that we will use there's a long uh web page here on fine web and they go into a ton of detail about how they process the data which is really fascinating reading by the way and I would definitely recommend if you're interested into Data mixtures and so on and how data gets processed at these scales a look at this uh page and more specifically we'll be working with the fine web edu I think and it's basically educational content from the internet uh they show that training on educational content in in their metrics um uh works really really well and we're going to use this sample 10 billion tokens subsample of it because we're not going to be training on trillions of tokens uh we're just going to train on uh 10 billion sample of the fine web edu because empirically in my previous few experiments this actually suffices to really get close to gpt2 Performance and it's um

simple enough to work with and so let's work with the sample 10 uh BT so our goal will be to download it process it and make sure that our data loader can work with it so let's get to that okay so I introduced another um file here that will basically download Fine web edu from huging face data sets it will pre-process and pre- tokenize all of the data and it will save data shards to a uh folder on um local disk and so while this is running uh just wanted to briefly mention that you can kind of look through the data set viewer here just to get a sense of what's in here and it's kind of interesting I mean it's a it basically looks like it's working fairly well like it's talking about nuclear energy in France it's talking about Mexican America some mac PJs Etc so actually it seems like their filters are working pretty well uh the filters here by the way were applied automatically using um llama 370b I believe and so uh basically llms are judging which

content is educational and that ends up making it through the filter uh so that's pretty cool now in terms of the script itself I'm not going to go through the full script because it's not as interesting and not as llm Centric but when you run this basically number one we're going to load the data set uh which this is all huging face code running this you're going to need to uh pip install data sets um so it's downloading the data set then it is tokenizing all of the documents inside this data set now when we tokenize the documents you'll notice that um to tokenize a single document uh we first start the tokens with the end of text token and this is a special token in the gpt2 tokenizer as you know so 50256 is the ID of the end of text and this is what begins a document even though it's called end of text but this is uh the first token that begins a document then we extend with all of the tokens of that document then we create a numpy array out

of that we make sure that all the tokens are between oh okay let me debug this okay so apologies for that uh it just had to do with me using a float division in Python it must be integer division so that this is an INT and everything is nice um okay but basically the tokenization here is relatively straightforward returns tokens in mp. un6 uh we're using .16 to save a little bit of space because 2 to the 16us 1 is 65,000 so the gpt2 max token ID is well below that and then here there's a bunch of multiprocessing code and it's honestly not that exciting so I'm not going to step through it but we're loading the data set we're tokenizing it and we're saving everything to shards and the shards are numpy files uh so just storing a numpy array and uh which is very very similar to torch tensors and the first Shard 0000 is a Val a validation Shard and all the other shards are uh training shards and as I mentioned they all have

100 million tokens in them exactly um and and that just makes it easier to work with as to Shard the files because if we just have a single massive file sometimes they can be hard to work with on the disk and so sharting it is just kind of um nicer from that perspective and uh yeah so we'll just let this run this will be probably um 30ish minutes or so and then we're going to come back to actually train on this data and we're going to be actually doing some legit pre-training in this case this is a good data set we're doing lots of tokens per second we have 8 gpus the code is ready and so we're actually going to be doing a serious training run so let's get P it back in a bit okay so we're back so uh if we LS edu fine web we see that there's now 100 charts in it um and that makes sense because each chart is 100 million tokens so 100 charts of that is 10 billion tokens in total now swinging over to the main file I made some adjustments to

our data loader again and that's because we're not running with uh Shakespeare anymore we want to use the fine web shards and so you'll see some code here that additionally basically can load these shards uh we load the um un6 numpy file we convert it to a torch. long tensor which is what a lot of the layers up top expect by default and then here we're just enumerating all the shards I also added a split to data load of light so we can uh load the split train but also the split Val uh the zero split and then we can load the shards and then here we also have not just the current position now but also the current Shard so we have a position inside A Shard and then when we uh run out of tokens in A Single Shard we first Advance The Shard and loop if we need to and then we get the tokens and readjust the position so this data loader will now iterate all the shards as well so I Chang that and then the other thing that

I did while uh the data was processing is our train loader now has split train of course and down here I set up some I set up some numbers so we are doing 2 to the 9 uh tokens per uh per um per step and we want to do roughly 10 billion tokens um because that's how many unique tokens we have so if we did 10 billion tokens then divide that by 29 we see that this is 1973 steps so that's where that's from and then the GPT three paper says that they warm up the learning rate over 375 million tokens so I came here and 375 E6 tokens divide uh 2 to the 19 is 715 steps so that's why warm-up steps is set to 715 so this will exactly match um the warm-up schedule that gpt3 used and I think 715 by the way is very uh mild and this could be made significantly more aggressive probably even like 100 is good enough um but it's okay let's leave it for now so that we have the exact hyper parameters of gpt3 so I fix that and then um that's pretty

much it we can we can run so we have our script here and we can launch and actually sorry let me do one more thing excuse me for my GPU I can actually fit more batch size and I believe I can fat I can fit 60 4 on my GPU as a micro bash size so let me try that I could be misremembering but that means 64 * 124 per GPU and then we have a gpus so that means we would not even be doing gradient accumulation if this fits because uh this just multi multiplies out to uh the full total bat size so no gradient accumulation and that would run pretty quickly if that fits let's go let's go I mean if this works then this is basically a serious pre-training run um we're not logging we're not evaluating the validation split we're not running any evaluations yet so it's not we haven't crossed our te's and dotted our eyes but uh if we let this run for a while we're going to actually get a pretty good model and the model that might even be on

par with or better than gpt2 124 M okay so it looks like everything is going great we're processing 1.5 million tokens per second uh everything here looks good we're doing 330 milliseconds per iteration and we have to do a total of uh where are we printing that 1973 so 19073 times 0.33 is this many seconds this many minutes so this will run for 1.7 hours uh so one and a half hour run uh like this and uh we don't even have to use gradient accumulation which is nice and you might not have that luxury in your GPU in that case just start decreasing the batch size until things fit but keep it to nice numbers um so that's pretty exciting we're currently warming up the learning rate so you see that it's still very low one4 so this will ramp up over the next few steps all the way to 6 e Nega uh 4 here very cool so now what I'd like to do is uh let's cross the T and do our eyes let's evaluate on the validation split and let's try to figure out how we can

run evals how we can do logging how we can visualize our losses and all the good stuff so let's get to that before we actually do the run okay so I've