Python and AI Power-Up Program Offline Class-20250821_190738-Meeting Recording

August 21, 2025, 1:37PM

2h 13m 5s

Margi Varmora started transcription

Tarun Jain 2:13

No.

Put a mic on now.

OK, so first what you can do is you can just create the copy of this in virtual environment. OK, so today what we'll do is we'll directly jump into the actual packages that many people use this. So we'll start with tick token which is maintained by Open AI and then we'll.

Probably proceed with ugging face, right? So ugging face ugging face.

OK, so in simple words, ugging face is GitHub of AI models, right? So how many AI models we have? Text-based related bots are a model Sir, right? Mr. Logaya, Quin Novaya. So most of the open source models that you have right, which is open weights.

They say GitHub a code. We say all the open source models that you have, it's saved on again face, right? UI checker. Let's suppose you have text to image is 1 model right?

Flux midterm.

So mid journey what OK you enter a text and send that image so that is stable diffusion. So just like that large language model is for text video images then you have stable diffusion.

So there you have models like Flux, right? Flux is open source. You can do it on hugging face, right? So that is the hugging face library and once that is done, we'll get inside embeddings, sparse embeddings, dense embeddings. This is mainly used for search.

Right. So yesterday we looked into tokenization, but we didn't actually implement tokenization, right? So today we'll actually implement tokenization and also understand how BPE works, right? At the end of the day, BPE is the core tokenizers that open AI users, all the open AI models.

Whether it is GPT or O3 series, O1O2O3 and O4 mini, right? And then you also have

one more tokenization which is called Wordpiece. So Wordpiece is developed by Google's team, right? So these are two of the tokenization that we will be using today.
BPE.
Byte pair encoding.
OK, screen disconnect or shareholder?
OK, so everyone has this notebook, right? So what we can do is we can just create the copy of this and there is also one note from Andrew Karpathy on what tokenization does, right? So let's suppose sometimes you think LLM didn't give you the right spelling, right? So then.
That's the issue of tokenizer, because tokenizer never saw that word before. So if a word is not seen before, what is the token that use it uses special token?
UN unknown, which is UNK, right? So we will try to use unknown in most of our use cases, but here there are also some of the areas where you think tokenization is actually a culprit, right? Whenever something wrong happens in LLM translation, right? Again, tokenization is an issue because some words you don't even know.
It exists. That is LLMS, right? So before we can jump into it, yesterday there were a few issues that we faced when we were running some quiz, right? So here I wanted to show this example again. In this example, do you think you'll get an error or not?
What you said it should have comma yes, so you should get an error, right? But getting an error last time yesterday, when did we didn't get the error? This was the question that we tried yesterday, right?
If you notice the difference between this example and this example, Atyantik is a complete string. It's not a single character. Google is also not a single character. So in threading arguments is a tuple, right? So every single.
Word that you see it goes as a character when it is in threading. So basically when you try example here it is just one single alphabet in the quiz that we took yesterday, right? So basically this A is just a single character. Even if you don't give a comma it will work.
Only if it is a single character, but if you give a link.
Now this link is not a single character, it's again a complete string. So now if I run this example, what should be the output?
Starting Google starting. No. Have I given a comma here?
This is an error. It is not a single character, so this is a string, right? So this will be an error.

So now if I print Python example one dot PY this is an error but for the same example if I just give A&G.
Without comma this will work.
Because it's a single character.
It makes sense. Now if I give AA or AT and GO, this is an error.
OK.
It is taking a tuple. If it is a single error, then it's going just as a single character. So when we give arguments, what is currently happening is it's like K comma T So if you notice when we get the errors, it says it accepted 2 arguments. Sometimes you will get 3, sometimes you'll get 5, sometimes you'll get 8.
That means this is going as a tuple. It's like a comma T So now a comma T you can't unpack the second character, so that second unpacked character is causing the error.
So yesterday when we saw this.
Who is directly we everyone we thought, OK, we didn't give comma, comma is the first part correct? So if I do comma now this goes as a single event. You can call it as a event loop.
So now if I run this as you see thread 80 starting same if I do for Google it will work.
So this was something that we missed yesterday and I was just checking with three different examples.
Move back and if you just simply give argument equal to single string a, would it work?
Single string ha this will work.
Without the like it's here. No, no, this won't work I think.
So if I give VA, OK, this works. So there was one more thing that we could have done here. So we have something called as Lambda command.
Given this works Lambda command. So here you don't have to give comma.
So this should be targeting also probably.
So if you see a Lambda command, it's similar to Python. You give function under score. You're calling that function. How do we call a function? Function name and argument. So you just have to give Lambda and then you have to call it. So here you don't have to give comma.
So if my function needs two arguments, you'll give two arguments. If your function needs one argument, you're giving one argument. So this was 1 alternative.
Then it is single character OK and and it is this bracket string. It just ignores the wrong bracket. It just ignores. It does not know whether it is a tuple or it is good

string. So it gives comma. It says OK this is a tuple.
So it's like this P comma Y.
So basically usually tuple is closed inside curly brackets, right? So what Python has done is I'm in parenthesis. So what Python has is even if you don't enclose that in parenthesis, you can write in this way. So since we have these two syntax, there is some issues.
String is immutable. Same thing. Tuple is also immutable.
String and string is basically converting into sort of. So one question. So if we do this everything in as in take take an input input as a tuple. So will it be the same or which one if we take an input for tuple OK and in in that input also do we have to part with the for my end?
If you're taking input as a tuple, here input is tuple only, right? This is tuple.
In normally to take will be with.
I didn't understand.
What would be saying we have tuple if we are taking tuple as an argument in in some other places?
I still didn't get.
OK.
So this was one thing and second thing. So basically the embedding I said yesterday it is actually used in embeddings, but with specific embeddings it's called word to vectors.
Actually I had searched this yesterday, so you have Globe. So initially when Stanford used to build embedding models, right? So we didn't had open AI embeddings initially, so after tokenization.
We usually go with embedding layer, right? So when we train any neural networks for sentiment analysis, we have to give an embedding layer first. So the process is your raw data. After you have your raw data, you tokenize it.
After you tokenize it, you go to embedding layer. First the embedding layers we had was like word to vector glow right? During that time we had post tagging internally used so that you can map. The best example yesterday we took was.
The royal Kingdom and as well as the animals. The second best example is Apple. If you remember we took Apple right organization. So Apple can be considered as object, it can be considered as noun. So during that time in Globe that targeted the problem statement like.
If there is any word, how do you actually classify them right? Whether it is falling

under this category or that category in the sense noun or a verb. So during that time POS tagging was used for this particular model. Globe embeddings, Globe and word to web.
But once BERT came in, which is the Google BERT model, this embedding model started reducing.
OK, so the reason why I'm sharing this POS tag again is because I actually want to show a workflow on how things will move. So this is very important. I just want everyone to pay very close attention to this. So let's take an example.
So what are we supposed to understand today? We need to understand what is tokenization, right? So what input did we pass yesterday? When we used that tool, right, we gave 3 words. So when you are using any existing solution, that model has to be already trained.
Right. So let's suppose now we have our own data.
Right, so you have your raw text. So basically what will happen in raw text is you'll divide your data into two parts. One is training set and one more is testing set. Every time when you join any Kaggle competition, build any ML models, you'll have two set of data. One you have to train the model and one you have to use it for months trained how to test it.
Inference, right? So when you're building tokenization, you only have to tokenize on training data, not on testing data, right? So now we have raw data and then we are performing BPE tokenization. How this works, we'll cover next.
And once you have BP tokenization, what will happen next is you're creating token ID conversation. So ML models doesn't understand text, it only understands vectors. So now what will happen is you have your broad text.
After you have your raw text, you're tokenizing it. So what will happen in tokenize? You're splitting it. Once you split into tokenize, you need to get these IDs.
So these IDs are nothing but vectors, right? So now once you have these vectors.
It will be passed into embedding layer. So understood embedding layers will not take your text data. Embedding is also a model. So whatever tokens that you get from tokenization, you will pass it to the embedding layer.
And once you have your embedding layer, then you have your model, right? So this model can vary. Is it a neural network based model or is it a transformer based model? So what is transformers? GPT, right? The T is a transformer and then you have BERT.
So BERT is a transformer. Yesterday we saw one more example. Robota. Robota is a

Transformers. T5T5 is a Transformers. So all these Transformers differs. Some are encoders only, some are decoders only, and some are encoder plus decoders. Right. So once you have your embedding layer, then you decide how you have to pass. So here we are looking at Transformers. In Transformers basically you have positional encoding. So this is not required for developers, but if you are into AI as a whole, if you want to get into research.

On how transformers works, it depends on positional encoding. If I want to give example by this is used. Let's suppose I want to convert 1 text into Hindi, right? So there might be two ways I can convert English into Hindi.

One is you have in straight order and one more it is in reverse order and both make sense sometimes. So if you want to know in what position the text has to go, this is where we usually calculate positional encodings and then you have your axle model which is transformers.

But the main thing what we need to understand is only these two things.

How do we tokenize our data? And once you tokenize the data, how do we get the ID's? And once you get the ID's, how to create an embedding model? Is this clear? This ID's that you're getting, where are these ID's map? It's vocabulary.

So basically here, right?

When you are converting this DP tokenization, you have your own data, right? So once you start your own data, if you remember yesterday, padding is a zeroth ID, UNK is a first ID. So every time you see one, that means hey this word is unknown. Every time you see zero, that means it's a padding token, and every time you see cat, probably it will be some X token. So how is your vocabulary mapped? It's just ID. It's like serial number that you are giving for every words that you have.

Created your vocabulary, right? So that vocabulary is created here first in BP. So let us understand this process.

So now what we will do is we have two things we need to understand. How are we generating that IDs and in what ways we are generating it and what is the limit of it, right? So in simple words you can just think dictionary as a sorry vocabulary as a dictionary, right? So here I have one example.

So this example is how do you convert a list of sequence that you have. This is a list of sequence.

So the goal is let's consider you are given a data. In practical also we will see what is the first step. The first step is you need to have your input, so I'll just copy this.

And I will paste it here.

So instead of example I will give this as input equals to.
This particular sequence and what is the type of it?
List. OK, so this is the first step. Now what you need to do is you need to start with BP approach. So it is like start with vocabulary of individual characters. So what does this mean? Usually once you have your input sequence right?
The first step is you need to convert it into character level tokenization. So what is sentence level tokenization? Once you see full stop, you have to stop. What is word level after space? What is character level?
Each and every single character, right? So now.
This is step one.
This is step 2.
What are the characters that you have?
LO.
So let's just check that. No, this is also I'll.
You can't ignore.
OK.
And just like that you have till.
OK, so this is the second step. What are you doing in the second step? You're creating the character level tokenization. Now coming back to the third step, this is the most important step, the 3rd and the 4th step. I mean the 2nd and 3rd. So what you're supposed to do now is you need to count the pair of the adjacent symbol.
So what does this mean? Whatever character level that you have, you need to create the pairs of it, right? So here you have a concept of N grams, which is do you want to go with bigram, trigram or foregrams? Usually people go with bigram, right? So now we have to create pairs.
Step three is create the pair of occurrence.
So in simple words, if you want to understand the difference between BPE and word piece, BPE is basically based on frequency, right? So what are we doing it here?
You're creating a pair and then you're just counting it. So what does count mean?
Check the frequency of occurrence.
Right, so let's create pairs. How many times is a low occurred?
123, right?
And what is the next pair OW?
OW how much time is OW occurred and note this should be a tuple.
How many times has OW occurred?

Then what is next?
Hello OW is done. Then again you have next is WE.
If W is not considered, WL will not be considered. It's a different sequence now.
Which one?
No, no. But if you're doing character wise LO, then OWA, then WL will not be there.
No, usually it's on the adjacent. Adjacent means you'll go with a different sequence only.
So basically, probably here you'll have a stop.
Invisible No. So when you are splitting characters split. So if you see start with vocabulary of individual characters. So when you start with individual character, every single sequence will be in character level. So we are doing board wise character level.
Yeah, we are first splitting by words, huh? Then we are doing character level, huh? So we are not doing altogether characters. Yeah, correct. We are splitting by words and then by character level. So that is the reason WL does not exist together. Huh. I just sent it won't come.
In terms of adjacent, WL will not come.
Now what about WE?
123.
So.
And which is the next pair? Uh, Uh.
ER is 1/2, again 3.
W is already there. Next we have ES.
Yes is 1.
And after years we have St.
BW's.
WW is done. ER is done.
Yeah, yeah.
I think that the TR is already there, which is 3. So now once you have your occurrence, the next step is you need to merge it. So how do you merge it? Whichever is the most frequent pair, we have to start merging it. So this will continue until you have number of vocabulary size, right? So let's.
Just suppose my vocab size.
Is 100 right? So you can only accomplish 100 tokens which is dictionary. After that it will stop. So here now we already have 1234567891011 right? Now how you have to start is you have to start with merge rule.

Merge.
The most frequent pair.
Which is the most frequent pair?
So LO will merge. So basically here you have sequence right? Instead of L comma O now you will have LO.
Understood. Now if you see it's a pair. So here also if you look at the tool, right?
So this D didn't find any pair probably in the data that it was trained, so it is marked as a different token. So there is no pair of D, so this step will repeat.
Now what is the next? This is merge rule one.
We have to iterate over everything again. Huh. It will be like this is probably 2 follow up children here.
Yes.
And then you have merge.
Rule 2. So first merge rule, what did we merge?
Just L&O. Now what is the second merge rule? OW, right? So you start with LO.
And then you'll have OW and just like that you'll have the remaining pairs LOOWE comma R Now W is alone.
Which one? So it's like you'll also have this space as well. You'll also have well. So there is a parameter apart from vocabulary size. You have one more parameter called min frequency.
Min frequency you can define one. So if you define min frequency as one, let us suppose I have LO, my data is trained and now if I say LO, if LO was not at all mentioned in my data, it will considered as a unknown which will be.
H comma E comma L comma L comma O. But now if my data already saw hello even once then it will be like this.
So I'm showing on the inference level how it works, right? Say let's suppose you train some data, you saw some new keyword.
This keyword doesn't exist so it will be in a character level. But if this keyword was even mentioned once, it will be the complete word. So here whatever I'm doing is based on the merge rule. So whatever you have previously it is already there in your.
Vocabulary size. So vocabulary size is a dictionary. So what is a dictionary? It's mutable. So here you already have the idea of it.
So if I show this diagram.
If you see, every single word will have its own ID, right? So this words already have their IDs. So now you are just creating new merge rule and you're creating a new ID

of it.
OK.
Merge rule in the sense this one only you are merging this thing right? You merged OW, you merged LO. So this will continue until your vocabulary size is reached, which is 100. So once 100 is reached, you have your vocabulary size.
So in practice this will be somewhere around 30,000.
And minimum size, usually it will be one or it will be two.
If your data is less than 100 MB something, if your data is more than 100 MB, this minimum size will be 5.
Hi, it's zero. By default it is 0. By default it's 0.
But that will not make sense. Let's suppose you give maximum as 10, right? And in your data yellow was repeated 9 times. Now if you define your minimum size, minimum frequency as 10, not even a single word will have. Let's suppose yellow is there, right? Hello will be added like this.
Even though you have nine times hello in your data, when you're checking the tokenization, you will have 5 tokens instead of 1 token. So if that's the reason why you keep your minimum frequency to be very less. If your data is like less than 100 MB, you keep it one or two.
Usually it will be two, but if it is more than 100 MB then you can have three or somewhere between 3:00 to 5:00, but I have not seen anywhere more than five.
I guess it is a maximum five or five only 5.
But by default it is 0.
I see a bit confused about the molecule.
So merge one. What we'll do is we'll see in the code as well. But technically what will happen is let's suppose you have eldo, right? So let's define the IDs.
Vocab size.
Of 0 is unknown.
Is this fine? So by default you are defining your zeroth index to be unknown. So every time you see a unknown keyword it will be UNK. Then you have vocab size of one.
To be pad.
So I'll just reserve starting five to six to special tokens. Huh. Which one? Yesterday what we saw. Yeah, yeah, this is configured. This one you have to set when you define or train the model, right?
So now I'll just leave starting 5 to special tokens. I'll start from six.

Your six will be L.
This is where we will create it from our own. I mean like the list of whatever data you have right now here when it is 6 next time when you ever you see this thing right L that will be.
No, it's already there in dictionary, right? So vocabulary size, whatever this thing is there, right? This is same. Let me show you. One second, I'll show one example.
So let's suppose I have open.
OK, now if I give space and if I give get and then if I use open, can you see different words here different IDs?
Different IDs. Why? Because there is a space here, but now if I give if I remove this. Can you see the same?
So The thing is, when you give space, it is not usually used in your vocabulary size, but it will consider it as a new ID. Why is this happening? It's because you're doing it on a character level.
If you give a space before the first open, it is going in the same thing. So when the first open give a space before that no. So here if you see it'll be same. If I remove this now it will be different.
Same goes for open AI. Open AI here AI is different is a big I take this point but the more through that we are doing.
I'm in the impression that you're going step by step. So you already combined hello. So hello has been paid. Hello is already there in your ID. Then how come OW will come?
No OW in the sense you have that on character level rate because that occurrence is still there.
So let's suppose you have yellow. No, it will create every single merge rule will have a new this thing.
It will create a vocab. It will create a new vocab. If not, let's suppose if steps this thing is there, right? You mean this one? Yeah, this one. This won't create vocab. This will create vocab merge rule.
Merge rule is just creating new tokens. This creating 2 characters. So OK, this step three, it's not creating any vocab. This is just counting the frequency, right? We counted the frequency. We counted the frequency. Now in merge rule you are creating a new word, right? So here now you have vocab size 0.
Six is a low then vocab size.
Of seven is West.

Right it is not OW because we are still at merge rule one right? So just like that go till end till R now.
Now your vocab size of let us say 11 or 12 is R Now what you are trying to do is you come with a merge rule 2. Now here you have vocab.
Size of 13 is OW. Which one? No single L is already there in your vocab size, right? So if you understand the rule, you have vocabulary size dictionary.
In vocabulary size dictionary, you are W won't be there because it would have been added. Yeah, I mean here I showed because we were going on the order, right?
So let's suppose you're starting from fresh list. Fresh list will have vocabulary size starting few of them as special tokens. Then you're starting with individual characters. Right. Once you save individual characters, let's suppose you came to vocabulary 15, right? Till 15, everything is a single character. Now your next token, which is 16, will be yellow, right? And then you will go with the merge rule next, which is.
OW. Here we are only dealing with two pairs. In most of the cases here you'll have three pairs.
Those those three pairs will have LOW. Now LOW will be in your some 30th or 40th. ID right? So every single time once you saved in dictionary, it's done. This will not create any pairs. I thought step three meant for merge rule. Merge rule will create new tokens. Here you are just counting.
So BP is mainly used for counting of your occurrence which is happening at this step. And this pair is decided based on north grams.
No. So this is dictionary vocabulary. Memory is different, which is our context vendor. The the space will be separate to open and then open will be sealed again. No.
Which one? This is not a separate token. It is considering it one whole thing. No wait. Why is space even a token? Space is never token open.
What was that?
OK, here we don't have that ID.
This one.
So space is not open, so.
OK, I'll come with the answer tomorrow. Solid answer, but I do have an answer. I just want to be sure, but I'll come. It's something called as contextual tokenization, but how it works.
Contextual tokenization.
So I'll come with this answer tomorrow, but it has to do something with this keyword only contextual tokenization.

It will be same for all the tokenizers. Which one? It will be the same case for all the tokenizers. This is there for word piece BP or if you use unigram. This is a common thing, but the only thing is when it comes to BP it is based on frequency of occurrence.
When it comes to what is this word piece, it is based on probability. So you will check the probability how much time is this pair already occurred before, right? There is a likelihood formula for it which we will see once we come with implementation.
Marginal will differ.
So if where is this? So if you look at word piece.
What we'll do which one?
No, Marjul, you're not doing it. Tick token is already taking care of that, right? Agging face is already taking care of that logic. The only thing what we are doing is we are just reusing the libraries, but I'm just showing you how BP actually works.
So here if you see word piece is bit different. Rather than calculating based on frequency, it will calculate based on the probability like this pair how much time it has already been occurred right? So frequency of first element.
I will just show one example, but let us suppose you have a pair called yellow, right?
So what is the frequency of this pair 3?
And then how many times is L occurred?
How many times is L in this sentence? Three, right?
And what is the formula into? How many times is O occurred?
So this will be the likelihood likelihood of yellow.
How much is this 3 by 9?
0.333 Now what about LOW right?
4W is also three.
This is 3 W how many times?
0.2.
0.2 right? So if your numerator is more probably, I mean compared to others, this will always be high. So we'll go with the last values. We'll go from last. Let's suppose I pick SP, right?
How many times has St. occurred? One. What about S?
Just one time.
This is one. This will be merged first. This is the merge rule of word piece which is based on likelihood.
And I'll tell where this is used.

And most of these models are there, right? Google related models, BERT, T5, those users, Word piece, right? And all these people have their own libraries, right? Open AI have Tik token. Google has something called a sentence piece.
Sentence piece.
So there is a complete library. If you see Google has released this sentence piece.
So here they use this word piece.
And what we are doing today is open AI kicked open.
So this is what we will be using. The library name is Tick Token and if you just check the first line, Tick Token is a fast BPE tokenizer for use with open AI models, right? So we'll use this library. Can you create the copy of the notebook?
And here I'll open one file which is peak token.
And inside tick token you have model dot PY and here if you see you have all the model names. So you have O1, you have O3O4 mini GPT 5, then 4.54.1. So if you see till GPT 40 they're using the same tokenizer.
So this is the name that they have given for their tokenizer Jason file which is O200K under score base.
And in the initial version of GPT 3.5 and GPT 4 they had CL 100K base.
Is this clear? And we will also use GPT2. So basically GPT2 what they do is they also consider space as a token, but all the models that we have now which is GPT 50103. From CL 100K, all the models will remove the spaces. So let's just check the difference and understand how to convert our data into tokens, right? So this will be useful. Let's suppose you're building any application where you're using Open AI. Before you can give to Open AI, you can just.
Do a check how much tokens I'm utilizing every single time. So that way you can keep track of what how much input token I'm giving, how much output tokens you're utilizing. So it doesn't require any LLM call, it doesn't require any additional package. So Open AI itself have this stick token which will be able to keep track of it.
So have you created the copy of it?
Now tick token. If you want to install it, what is the command?
Tip install tick token.
In Colab it's already installed, but if you want to install the upgraded version of any package, what you can do is you can just do pip install then space hyphen you space tick token.
The reason why I'm showing is in Colab it's still using GPT 401, right? It's still upgraded till there, but in recent days you also have GPT 5. So if you tell to use a

tokenization of GPT 5.
It will tell an error. Hey, I don't know this keyword, right? So what we will do is we will directly give the model name. So what is the tokenizer model name? What is the model name here for GPT 5?
Or 200K base.
So copy this one. OK, I guess you don't have this file, but.
Here I will define GPT phi
Token model equals to 4200K hyphen base.
Just like that, I'll just add GPT.
4.
Token model.
What is the model name?
CL 100K base.
OK.
For GPT 2.
Token model.
You can just define CPT2.
And once you do that, create a new code cell.
Don't try it after text, just create a new code cell.
Here you can add ENC one or I'll just tell GPT.
To token.
Equals to.
Tick token dot.
OK.
GPT 4 token equals to tick token.
OK.
M.
So this one will.
Does not merge spaces.
Will merge spaces and this one will merge spaces.
You understood what we're trying to do. We're just defining the model token tokenizer model name and then we are just reusing the tokens, right? And how do you reuse? First just define it. How do we define tick token dot get encoding?
Add the token name.
So for GPT2 it is a ZPD2, for GPT4 it is CL100K, then for GPT 5 it is O200K.

Now we are creating a new port cell. I'll just define user input.
Just add some additional spaces in beginning, then just write hello space.
And GPT, GPT2 token. You can just copy that and just use a function called encode.
So what are you seeing here? You're seeing 2/20/220/220 which is there are three space. So if I copy this here.
Uh, where is that tool?
Can you guys see this tool here? Everyone of you have this link right? Just click on that and remove this and just add.
After open.
Give us space, you will see 220.
And then if you give anything here again open, that space will be gone.
This is happening only in GPT based models which is 3.5 GPT4GPT5, but GPT2 still has token for space.
Can you try it for GPT4 and GPT5 just to check the results?
No, it doesn't convert it. It ignores. Space is ignored.
So if you want to decode this, you can decode by just passing this ID.
GPT4.
Model token dot decode.
Yeah, it added 256. I added one token 271 is 1460.
So here if you see space, hello is a new token.
Did you try that?
Can you give hello? Hello. Hello.
Or we can do one more thing. We will try the same example that we tried earlier.
Open AI is a big AI company.
So basically your AI and AI should be same when I try it.
So there are two same tokens if you see 17525.
This is 1 second.
It's AI and now if I use space here.
It's a different token, so probably 3464.
For GPT 2, two space is a token other than one space.
So it's considering space before the given character as one token.
The space with EI, but if I just.
But it won't be individual token.
So there is no individual token for space. So if I give 3464 this is space space big.
Yeah, I so for two two spaces there is a token. Huh. Two space there is one token. But

for GPD two it is separate, right? One space is 1 token. So if I run this.
1505.
OK, GPT5 not.
GPT2 token decode. I'll use the previous example.
So here if you see you have 123.
So every single space is 220, then hello is space hello.
So here if I if I give 2374.
It is space alone so that means GPT does not merge spaces but whereas GPT4 and GPT5 if there are two space it is considering one but for single space it is down so that means.
If I give AI and space AI, these are two different tokens because this is space.
GPT 2 and it will be different tokens. It will be different tokens. It's a different model.
Even for four and five will be different.
PCHPCI 2 tokens, yeah.
But if there is, wait, it should be 3 for GPD 2. Space will be one token, no?
Space AI space AI. No AI space AI. Yes, two different tokens for all three. But what GBD do also it is it give to if I use space.
Now you have 220. So this AI is taking space plus AI. Yeah. Then there is one space which is 220 and then AI.
And if I do this for GPT 5, what I think the only difference they did is.
It's like if there are more than two spaces, they're more than one.
No.
So what it does is it splits by first page. OK, so any word that is a previous page, the splitting is done and then if there are more than two species in anyone, it just combines to one space.
That is what is happening. Yes, yes. I I mean on space hello. The space hello is 1, one token, huh?
That is what it is in all greater than GPT 2. Even in GPT 2 all there are only two models after GPT 2. So from GPT 3 to GPT 5 plus the reasoning models which starts from O series all are using only these two models.
You want to the word.
The space before the word is given to the word, but generally to it does not then breaks the next spaces to it would have individual token.
So if there are two spaces, GPT will have two different.
Why that thing would be two tokens in DPD 5?

So here we can try one more example. I have just added some of the text. What we can do is we can just run this code and here what we are doing is we are utilizing tokenizer and then instead of using get encoding, what is the function we are using here?
It has to get the encoding from the model itself, right? So this is the second function. If in case you know the model name, which is good. If you don't know the model name, you can just define encoding for model. Here you can define GPT 4O. So what are the models? You have GPT 4O, then you have GPT 41.
No, they're different. GPT 4 O is 14.1 is different. That model is same. So the model is same for O1O3O4 mini O5.
Then 4.54.1. All the latest models are using O200K. The deprecated models are using CL 100K.
And the latest open source model that they released, GPT OSS, it's using some harmony.
So.
Which one? Which one? We cannot calculate the the program price. They didn't give it in the open source model. Ha ha, correct. Because open the then then it would be a problem for.
So now if I run this tokenizer you will see that model name which is encoding O200K base.
And now what you can do is you can just run this for loop.
And it will just decode which particular word is assigned to which token. So if you see here now root to enter sequence is not a single token. It is NAR is 1, UTO is 1. Now why is that stopped?
Because it will go to certain vocabulary size. Until then, if it found that entire word, it's fine. If not, whichever is the maximum merge rule, it will stop there. That means NAR was the maximum vocabulary size that it was able to accommodate, right? So it is NAR. It is stopping.
So how did we get this only three keywords? It's because of the merge rule. First you calculate the number of occurrence, frequency of occurrence of the pairs, and then you start the merge rule and till where will the merge rule continue?
Until it reaches the vocabulary size maximum.
So like it again, I didn't understand, right? What is it? Vocabulary. See, Naruto could have been a whole word. OK, what I believe now. Sorry, I have to. It didn't for so long. It could have been a whole word.

But there were so many frequent other words. So infrequently chart where it was sorted out, it came first and then there was a predefined dictionary size that it can only have a pelmery size, but can only have 3 million or 4 million, let's say something. OK, so all that permission and combination that was going through it got.
In the dictionary. So that's a dictionary, right? Dictionary. It's like you keep on upending. You saw a new word, you open in the ID.
It didn't add that word. It didn't even add the NARU, right? If not, it could have started that. The best possibility was it could have stopped at NARU because PO is a same token. So we could have saved 1 token here, but it didn't save it because there is no possibility of NARU.
Now it will be sequential, sequentially not. It will be based on the occurrence.
You can just cross check once like which word makes sense, which word doesn't make sense. So that way we can get to know OK here merge rule was actually visible right? In most of the cases all these words that you give right Naruto or Asangan there only will see the difference.
In other sequences like fluctuates, if you see fluctuates is different.
This is common. OK, gravitational. So there is a word called lensing. So lens is different and ING. This makes sense.
So loop if you see Lu is also not there. So that means L is a single ID in your dictionary which came from character level.
What do you call character level tokenization?
It can be puffy, puffy, huh? Puffy, puffy. So there is more occurrence too, right? So when there is more occurrence, it starts forming a word dictionary. If there is occurrence, it will merge and it will add in your vocabulary, Sir, which is your dictionary.
Who is that?
Look, just me.
So if I do GPT 4 om token dot, do they have vocab size?
And vocab. So if you see it is only till 50,235, 257 for GPT 2 token. For GPT 4 token it's till.
1,00,000 GPT 5 it is still 2,00,000.
So 50K then 1,00,000 then 2,00,000 then probably now if they release new model.
So they haven't for GPT 5, they should have increased it, but they use the same thing what they've used for GPT 40. Probably the next model will have 3,00,000.
No, GPT4GPT4 is CL, GPT4O and GPT5 is still same, so probably they should have.

They didn't increase it. Next model probably might have 3 lags.
So this means whatever you have here to, I mean the two lakhs are your tokens, right? And you can probably also check by printing it where you can save it into Jason and once you save in Jason, it will be similar to what we saw yesterday. Vocabulary dot Jason.
So here if you see you have this tokenizer dot Jason.
This is a very large file you will have for every single token ID what is the size of it. So here if you see begin of text yesterday what we saw we saw EOS and BOS. So for Lux Lama that we released our tokens are begin of text and end of text.
So this will differ. For Mistel it will be different. We might have used UNK as unknown only. We didn't want to give UNK. So these tokens will differ based on model to model.
For token for the this tokenizer, can we? You have to train that. Can we manually? You want to open it manually because manually is not.
No train in the sense. I'll show you how to train it.
Now we have the code how to train it, but you can't do it for tick token. You'll have to pay for that because this is not open source, right?
That I have to check because we usually don't train the tokenizers much. Either it will be your embedding, but if you are changing the language then you have to train. But this training will happen if you are on open source.
Huh. I'm not sure about the closed source. There you can the O200 Carmony.
Tik token fine tuning.
I don't think they support it.
Tick token doesn't support fine tuning as far as I remember, but if you want to fine tune BPE based on your own data, that is possible. At the end of the day you just need a Jason file.
If you are building your own language then it makes sense. Let's suppose VPT is not good in Gujarati and if you want to build Gujarati to English translation then you will have to fine tune.
So this you just have to read this. If you are facing any issues in this category that means how to fine tune tokenizers and.
So if you see why is LLM works at non-english language like Japanese tokenization issue. That means you have to train your tokenizers. No, I mean like financial data and it has some kind of a vocabulary of its own, but hardly it will be few tokens. For that you don't have to train a model, but for language it's like too much of tokens.

So you just have to check like how much new tokens you are doing financial. I don't think anyone fine tune it for financial. But if it is for legal judgments, let's suppose you take some Supreme Court's judgment that you have 50,000 documents. Each judgment will have some, but it will be English only.
If one of it is such documents, the language is English. The language is still English. It is going to repeat itself. It is just going to be some words that are different. So during that time what you have to do is you don't have to fine tune tokenizers, right? Then you have to fine tune model.
So why model? Because you have to change the tone in which you have to generate the response. So let's suppose you fine tune a model. It won't make sense because if there are 10 unique words, it will have no occurrence. It's not gonna make the dictionary either way.
So if you see here in this models right none of a single model we fine-tuned the tokenizers. So Panda coder it was coding related LLM. So code in the sense English. Why do I have to train a tokenizers? Buddhi also we didn't. Buddhi is just.
We increase the context length. So all these are still English, right? So when it comes to English, you don't have to fine tune A tokenizers. Only for the Lux lama we fine-tuned it because it was completely new language. These are model. These are LLM fine tune models. Huh.
If you want to change any tone right during that time, you fine tune LLM, but not tokenizers.
So here is 1 action you said something worse.
OK, so here is one actual usage. So basically what you can do is every single time when you think you're using LLMS in one of your application, before you can call LLM, just check how much input tokens you're passing. So most of the time what happens is if I'm building rag.
I have my own prompt, then I'm getting data from my documents, right? Let's suppose I ask what is work experience of Tarun? It will look into my CV, it will get some new context. So every single context is a chunk, thousand chunk, thousand chunk.
So now you have 7, usually use four to seven, that is 7000 words or 7000 tokens that you're utilizing with your input and then you also have output. So this is just for one single query, just like this one query, you'll run this for multiple user queries.
Every single time how much tokens you are utilizing. You can just count this and log it, right? That way you can understand how much cost you are utilizing for one single

LLM call, right? Cost plus token count.
So here you have some simple text and then you use tokenizer dot encode same syntax. Now what this will do?
Once you do encode, what will this line do?
You'll get the IDs, right? IDs are individual tokens.
And once you get that individual tokens, you can just print the length of it.
I think it is what the cloud AI get.
So you have 57. So usually all the traceability tools are there, right? AI gateways or guardrails. They have these things inbuilt observability tool.
The GPA you're calling, it will use that tokenizer to see how many tokens you have to use. How much tokens? It should be for both. Usually what observability told us is input tokens is this much, output tokens is different because both have different prices.
Oh, yeah.
So here what you have to do is let's suppose I pass input tokens in my input tokens. Now I have my simple prompt and I have context so that will cost somewhere around not cost. It will have somewhere around 4005 thousand tokens.
So now you just have to multiply 4000 by. You have to just do the math 1.250 it's for 1,000,000. Now for your context, how much does it cost? This is for input. Now you pass this prompt.
To LLM, LLM generates output. That output is this price. So now if you look at output pricing, it's $10. But input tokens are very cheap, so you also have to do the math of this. So before and after both calculation needs to be done.
This is only for one LLM call we are talking.
Huh.
You can, right? You can concatenate, huh?
Because now we are not getting the extra text or model validator, you can add more after and then you can define some logic till where you have to concatenate.
You also have Max tokens, but Max tokens usually never it reaches.
So here you can use pydentic for input tokens, right? Because input tokens you can try to optimize the context, right? Most of the time, if you saw here right, you have for spaces, you have one token, so you can remove spaces from your input tokens at least.
So that way you can have some cost reduced.
So cached input is basically you have a new argument.

This you can't try trace. It's cached on their site. Whether it's cached or not is up to them. No way they will give in response. By the way we have different for every single LM call who will cache it right? So cache is something that you'll have to do separately.
And this is just one argument. I guess you just have to define cache prompts and then you have to keep it true and then it will cache your responses.
Hi you have to when you define the client right open your client you have to define cache prompt. So this way if someone asks hi good morning this kind of tokens is already cached so no one will reply back for those things.
So which one? Which are common questions, right? And if you have your own data, most of the time you are fetching the same context. Those contexts can also be cached, so that is.
As per the cost, right? That much of amount will be used if you case it.
Now can anyone tell me how do I just get it till 4 decimal point?
So here if you see you have 00171, I want to stop till here which is at my third decimal point. So what function should I use here?
Round.
Then.
Comma you want to stop at 4th so four. Is this clear? You don't have to use math dot round because in recent releases it is just outside.
Before it was an under math.
What happened? So basically most of the multiplication, but even math is a very lightweight library.
I see that like, uh, working perfect. No, no, no, no. Toting point, which is something that you only find. Yeah, you do 0.1 plus zero point.
OK, so let's actually build how you can build your own tokenizer now. And there are two sample code snippet that we have. One is for how you can take sample data and then how to take actual open source data. The code will remain the same.
Since we are writing code for two times for word piece, you have to do it by yourself.
OK, so here what we will do is.
We will define how to train BPE with some level of context so that I can show you how this hello works, right? So if you see this sentence doesn't make sense, but I've added hello here, then again I've added hello here and somewhere here I've added models.
Because I just want to check how on inference level it will make sense why minimum

frequency should be 0, why minimum frequency should be one. Once we understand this example then we'll then we will go with actual data which is open sourced one.
Right, so since we will be repeating same code right, you will use VP2 times.
Word piece is something that you have to just test it on your end, right? How to fine tune or how to build a tokenizer using word piece. I'll tell the logic, but this will be our first assignment. Is this clear? OK.
So now we will be using ugging face. So far whatever we used was open AI's library token. Now the alternative way, as I said, ugging face is mainly used when you want to deal with open source LLM or open source AI model in general, whether it is text based.
Image video. You have the ability to use all of them, right? So you just have to open hugging face.
So if you look at tagging phase and if I click on models.
This was the latest model which Gwen released. Gwen is basically popularly known for LLMS, but recently they released their image edit. So you can upload your image, tell a prompt what you need to edit in that image and it will edit it for you. So this is that model.
And.
If you want to fine tune it, you can fine tune it, but if you just want to test it, you have spaces.
Oh.
So this is free. What you can do is you can just test this. Here you upload the image, then you give a text and then it will display the output. So every single model will have the space for which user can test it.
So this is ugging face. You have all the models, text, image, video and they have at least around 10 Python libraries separate for text, for fine tuning, separate for video diffusion, separate and.
For tokenization it's separate. For embedding model they have a separate library. For every single details they have a separate library and all the libraries are probably more than 10K stars and the main one transformers probably it should be 1,00,000.
So this is their main package which we will cover once we start using open source LLMS. So if you see it's very popular library and many people use this library a lot, right? So if you are into open source, the only library that exists that works is ugging phase.
Huh. So he maintains diffusers.

So this is the library diffusers. No, he's into this team. Here is.
He does not give a talk at all. Not sure, only outside India. So if you see this libraries are there, right? Let's suppose you're using any library from mugging face. Probably we might not use all the libraries, but I'll just tell which libraries are useful.
I'm just imagining everyone of you are into LLM. Most probably you'll just use transformers. But if anyone is just curious, I want to learn about text to image, text to video, then you will go with diffusers, which is for stable diffusion if you pick any one of this library.
In compulsory you have to learn these two libraries. One is accelerate and one more is optimum because these are every weight models. You are running it on GPU if you want to have faster LLM response. If you want to use quantization, this is for quantization.
It's like 10 GB model, right? If you want to quantize it, you use optimum and if you want to have speed, use accelerate, right? So if you pick any one of this, this too is by default and if you want to build your own model that is, hey, this model is good, but I want to fine tune it.
You will use this one PEFT which is parameter efficient fine tuning which we will use in one of the sessions when we have fine tuning.
This is common. This we'll use now only data set, but the major libraries are the six and the optional one is.
Anyone of this?
But if you since most of you are into LLM, you have to pick transformers.
And these are very easy library accelerate and optimum. It's just two to three lines of code. If you use this library, how to integrate this with this library? That logic is written here.
So here what we can do is we can just define this line of code tokenizers.
We don't have to do pip install transformers. I'll add this in text.
And we use for edge line.
Oh, diffuser for like anything except.
Diffusers have different stable diffusion pipeline. Diffusers is a short form for stable diffusion. So stable diffusion is like text to image, text to video, image to video, those kind of things. If it is text to text, it is lugging face.
And image to text is also if it is one form of data to different form of data, it's diffusion, mainly image, video and audio.
So the first line of code here is we have to define tokenizer. Previously, how did we

define tokenizer?
Tick token dot you need to define a model name. So what is our model name now? So basically O200 base. If you open a tick token, all these libraries are what model? So O200CL200 then O200 Harmony under what model does it belong to BP right? So here since we didn't build our own model to name it, we want to define the actual model which is BP.
So just define tokenizer equals to.
From tokenizer tokenizer.
Then we have models dot.
If you see you have different models, you have BPE, you have Unigram. Word level is not that much useful. Then you have Word piece. The very useful ones are BPE which is by open AI and then you have Word piece. I don't know if it is DeepMind or Google, but it's Google.
Right, so you have work piece.
I'll pick BP.
And once you define BP, that's it. But the only thing is if you want to define a unknown token, you have to define that. So what is unknown UNK?
Is the function called dot VG function. This is instantiation. So inside this you will have unknown token. I just want to see if it is unknown or UNK.
Huh. It's UNK.
Token equals to.
You can just define a list and then write UNK.
And basically when it comes to tokenization, we have to follow 4 process. One is pre tokenization and once you do pre tokenization, what did we discuss yesterday? There were as soon as I showed NLP slide first.
I mentioned 3 keywords. One was tokenization. What were the other two keywords? Normalization and vocabulary, right? So what is normalization?
It's mainly used for preprocessing. If you have any white space, remove the white space. If you have lower case, I mean if you have text convert into lower case, stop words right? So here also you have a normalized function where mainly we will use white space so.
Pre tokenization and after pre tokenization you have normalization. Then you train the model. So what is training the model? You have BPE. I will pass my own data, I will train it and then you have post.
Post tokenization where you test it so pre tokenization.

Normalize, model and then post tokenize. So this four step we have to do.
So I will just define tokenizer dot pre tokenizer equals to.
Pretokenizer dot. If you see you have dot there are multiple things. You have split, you have byte level. I will just select white space.
So this is how you want to tokenize, right? Either you can do it on character level. The best option is do it on white space.
And as you can see, it returns none. So you're just what is this called?
So this is a an object and this is the attribute and attribute it will be optional. So what you're doing now you're defining it with white space.
Same what you can do is you can define tokenizer dot decode.
Decode is basically for post what you call post tokenization. So we have pre tokenization. We have defined the model. Here we are defining for post tokenization, right?
Encode and then decode. Decode is happening on post tokenization. So define decoder and then you have decoders.
Then you have BP decoder. Here you just have to define a suffix.
So what is the suffix?
Default is back slash W you just have to copy the same thing.
How we can keep it same as well. If not, we can just define suffix.
If not, you can just add it as suffix.
But since suffix is already W which is with HTML tag, we'll use the default one.
This whitespace and this whitespace whitespace till here everyone are completed.
Can you just cross it? We define the tokenizer where we define BP inside BP. What are we supposed to define our unknown token and what other tokens you have seen?
5.
Here we'll use pad. Unknown is already used. We'll also use something called as classification, which is CLS and separator, which is SCP.
When we define the trainer, we don't have to give the pad. We don't have to give. So pad token only happens when you see the data, right? Right. Once you know the data then only we can add pad token. So when trainer is defined during that time we can define special.
Special tokens. This is again how the library is built. So usually everything should be at special tokens level, but the added UNK in the model definition layer.
So can you see this? You have special tokens here.

So in special tokens also you can define UNK but we will not define why? Because we do not want to repeat it. But technically special tokens is defined in one uh what you call one argument only. So what special token we have we have UNK which we will remove.
Which is unknown. Then CLS is for classification, SCP is for separator, PDPAD is for padding, masking is for the model. So if you learn transformers right in transformers you have multi head attention.
So in that multi ad attention you have mask.
And by default it has given some value for minimum frequency, right? So that minimum frequency is 2.
Oh.
But by default it is.
0.
And I don't want to have 10,000 because I have very less text, so I'll just keep it 10,000.
And then just define trainer.
I'll just remove this.
Just crosscheck if we are on the same page.
We defined a model, we defined pre tokenizer, we defined post tokenizer and then what is the 4th step? We are defining the model.
And now we just have to train it and test it.
Is it done? OK, so now can you see this using simple list of data? There is some already existing text that we have. What is the type of this?
List right? So when we use our own data during that time also what will be our input? What will be the input if we should have our own data?
So if I want to start with VPE, what is the first step? Type list, right? Even when you bring your own data, you just have to ensure the entire thing has to be inside a list.
So I'm just running this cell. It should be list of strings on list of words will loop it through each and every string. So obviously it will be words string only, right? If you use single single words, probably doesn't make sense to train a tokenizer.
So you'll have sentences of some dialogues, which is in Hindi, which is in Gujarati. Ha string.
No, it won't take. It will throw an error.
Maybe we can try, but usually it is list of list inside the strings because float you can't train because you're training a model float will not make sense. So the only thing is

list.
And now what you can do is you can define tokenizer dot.
You should have a function called train train from iterator.
Text and then trainer.
Then.
It took less than one second, but technically when you have your own data, this will go at least two or three minutes, right? Here is just five or six text.
So now you can remove this two cell. You have some example text here. So what is its example text? Hello machine learning models tokenization. So what is my minimum frequency?
Two. So should hello be H comma, E comma, hell comma or should it be just hello?
Technically it should be lol. LO should be 1 token because LO is there twice.
Machine learning. You can't predict what it will be based on the model. Then models. I technically repeated models twice. So what we have to do is we have to test with LO and models.
Yeah, same event.
Can you add U1K here and then test?
Just add this UNK here.
In special tokens. Now if you see yellow is a single token, can you change minimum frequency to be one and run it again models if you see models is also.
Minimum frequency you have to make it one. This I will make it 1.
Sorry, not one. Make it 3.
Minimum frequency make it 3 because hello is already repeated 2 times. Even if you make one you will not see any changes.
Now if you see it is showing H comma El. If you are getting error you have to run the model all the lines you have to run.
Because.
See here what is happening. Are you assigning this somewhere?
Where so tokenizer dot train your whatever tokenizer you have defined is trained. So now once you change anything in your data, what should you do? You should run your tokenizer and everything at one more time.
So basically it's like once you train it, you just use it for inference, but if you change anything you have to run everything again.
So now can you see hello in one token?
No, no. I mean you kept minimum frequency as one or two. We made it three.

Hi, if it is true, it will be a low, so we made it 3.
Huh. So now it made sense. What is minimum frequency? I'll repeat again. Minimum frequency. If your data is less than 100 MB or something, keep it one or two. Two is good number if it is more than 100 MB.
Or G in GBS have it somewhere between 4:00 or 5:00.
Having more than 7, it's probably it will degrade your performance, right? Just stop it somewhere less than 7.
Now I will not write the code, I'll just show you how to use the data. You have to write here the code by yourself. So what will be the first line of code here?
So now you have the data, you have to train BP. So how will you start?
Tokenizer equals to models BP unknown token then.
Pre tokenizer, then post tokenizer.
Uh, which is post organizer and then?
Train. So when you train you have to pass a data. So how do we pass this? I'll show that logic here.
But you understood what to do in these three lines of code, right? How to get list? I will show that logic. So I have written these two lines of code.
Is this clear? OK, so now if you notice this keyword here data set, did I show this keyword before hugging face add Transformers, diffusers, data set, PEFF, accelerate and optimum?
Right, so Kaggle was probably very popular or in come to data set, but in recent days after LLM there are so many data set which is available on hugging face, right? So if you want to use any data set name, whatever you see here is a repo name.
Whatever you see here is a data set name. No, this is data set. So let's take one more exam.
Same huh. So if I open a game face, right? So what is the repo name? AI Planet. AI Planet. What is the model name? Panda Panda Coder. So name if I open data sets.
So how much data set do we have here?
close to 5 lags you can say. So now if I want to use this data, let's suppose I want to use awesome GPT or ChatGPT prompts.
I will copy this.
I will come back here, but where are we getting this data set from? When we installed which people installed with the tokenizer? Is it coming from? No, it's coming from data sets. We have to do people install data sets.
You're using VS code. There you have to do pip install data sets.

So now if you see it downloaded the data, but I want to show the dummy data as well own data.
Equals to.
It will go a bit slow here. BS code like a falling behind. Hi I'll I'll come back. So here we are just loading the data set. We are doing nothing yet still.
So here what should I give? What is the first thing?
And then.
So here I'm just showing one more data set. We already have one data set which we want to use in our code. I'm also showing one more.
So now if I click on own data.
So can you see you have ACT and then prompt? There are only 203 prompts, I mean rows. So if I want to cross check I can come here. How many rows do I have? 203. So what should be the syntax I have to pass?
Repo name and username this entire thing.
Is this clear?
This will take time because it's downloading the data. Huh.
So can you check where data set is loaded inside data set? It is saved there.
So I'll just show you how to debug this. How do I check the data?
Everyone are able to download.
Error.
Oh, probably you can just remove that and add the entire thing. Even this works.
So what you can do is you can directly add without comma.
You're able to download.
So either you can give it comma. If you are getting error then you can directly give the repo name. So what is the repo name? It should have repository plus the data set name.
It is 203 and if you come back here.
2:03.
But now can you print whatever variable you have used?
This is what you are getting. Now if you want to check the prompt, how will you get inside that prompt?
So let's suppose I want to see starting 5 prompts. What logic do we usually use in list?
So once we use list, right, I those two were our first concept.
That is called indexing, right? So now let's suppose this is my dictionary. I want to get

here. How will I start?
This will come when you do you open AI as well. In open AI your messages will will be inside messages, then inside messages it will be in zeroth index, then it will be inside content.
Own data no after own data train. Why train? What is the type of this if I want to use dictionary?
What should I use? Key name? What is the key name here? Train inside train?
You can directly use prompt or it will be features. Let me just check this. It is train.
And then it should be prompt.
So if I remove this.
I'm directly getting the text.
So now if I do the length of this, how much should it be?
2:03 So you can check now what is the data in this 0 to 6.
You got the logic how to debug and check what is there in the data. The best option what we can do here is.
Own data.
Oh.
Pandas.
What train was a data set which has features and numbers?
Train as a this own data and then train. Yeah, this is a data set, correct? Correct.
Which has features and numbers, huh? We're not using features, no?
So since once you use train rate, once you define train, you already have two dictionaries. Those two dictionaries are at and prompt.
OK, so here let's take DS. Everyone have run this DS.
So now if you run DS, you have training data, you have testing data, you have validation data and every single data has a key. What is that key or text? So how will I get the text of test?
Test, test and then text. But why? Like wouldn't wouldn't it be another one? OK, you mean you want like this features, right? Right.
And then text feature will give key error.
Oh, no reason.
But in the data setting, it should have the features. It should also have number if I want to move here dot feature.
dot features.
Huh. Now it makes sense if I do dot features.

Start text.
It's printing only data type. So if I again print text, if I run features when I run text.
In bracket if you like features I think you can with bracket. No no no in in as an add it.
No it's not working. No it with like this. No no no test then again add a key of features.
You wrote feature only, not features. Features then no no like in square brackets.
You mean like this?
No, no, no, no, no, no.
Open.
OK.
We tried with feature, not features, so this might work.
No features. You can't run that because when you run features it's only printing you the data type. It's not printing the actual data. So again I'll repeat here if you print here I'm doing dot features I'm getting text but.
At the end of the day, what is the text printing? It's just printing the value string.
What I need is the actual data in it, so I'm just doing dot text.
And now this is length of 4000 something.
Now what we need to do? What format is it in? If I'm able to get 4300, if I just print this, is this in a list?
But inside that, is it in the list? Yes, if I run a for loop, let's suppose for I in.
0 to 5.
Print I.
I'm able to print.
So this is what I need. I just have the list. I will save this list in a file. That file I will train it.
But you're able to understand the logic on how to use data sets in a game phase. No matter what data set you want to use, first use load data set. Once you use load data set, try to see does it have any testing data, training data or not.
If you see any data set which has less number of downloads under 1%, they will not have any testing data or validation data. If you check any data set which has more up ports, that means they will have testing data which is an actual data.
Right so here if you see you have test train and validation.
How much data you have in train? It's like 18 lakhs, right? So I don't want us to spend that much of time. So we'll go in validation or you can also use test like you have 4000 tests, we will use the use that for.

For training a tokenizer, is this clear how to check the data set?
You can try with other data set as well.
And there is one more command. So let's suppose I do own data or let's suppose I use DS. Then if I use test, after that you can use a function called to pandas. Pandas is nothing but a data frame.
Which will print your data in a tabular format.
So now you can check since here we only have one column, but in most of the cases you will never have one column. We will have 1020 columns. During that time checking that in a tabular format is much.
It makes much more sense and there are different pandas command. Yesterday you saw numpy right? Just like numpy you have pandas and matplotlib which we will cover once we start data preprocessing. I actually skipped it because we were on online mode. I mean once we are in online mode we will use that.
Clear. Is it clear? Two things you have to notice in load data set. One is once you use load data set you have to give a repo plus data set name, not just data set name. And second thing is either you can use pandas.
If not, you can go with.
Uh, the train and then the column name. Is this clear?
And now what is the first step?
OK.
Tokenizer equals to.
You can just copy paste. It's better to copy paste.
Same VP. Word piece is something you can try for Word piece. I'll send new data set and I will send. I'll send a new data set. You have to train Word piece so you can just have a new file.
In that new file you can just have word word piece and the data set that I share. So that way you will know what column to pass and how to train it. So it will become two tasks in one assignment.
Is this clear?
What happened?
And what is the second line of code?
Bing.
So let's suppose I don't define vocabulary size or minimum frequency now. OK, I'm just removing it just to show what is the default values.
Can everyone check if this four lines of code is done?

And this is not new. The only thing what we are doing here is we removed vocabulary size and minimum frequency just to check what is the default values.
Can anyone tell me the attribute or at least the value of vocabulary size if someone who has already ran this line?
Trainer dot then you will get that attribute.
Trainer dot you have vocab size. I'm just asking quiz.
No, I didn't define anything. I didn't define any default values 30,000, that's correct for minimum frequency.
So now what we can do is, is this data more than 100 MB?
It will be less than one MB because it's just 3000, right? So what should be our minimum frequency? One or two, right?
We can again do this thing.
Vocabulary size. I'm keeping it as I'll keep it 20,000 and minimum frequency either two or one. Anything works.
You understood? We actually didn't define the default values. I'm for vocabulary and min frequency. If you want to cross check it, we have to use trainer dot vocab size trainer dot minimum frequency. Same thing what you have defined inside.
In VS code.
Can you tell that reason why? Why auto complete is not working? What library they might have missed properly used in?
What is that library? No typing. They might not have used it properly.
Autocomplete will only happen when it has much context about what each parameters are.
These are the same four lines of code that we have already written.
Oh.
We just have three more lines of code to write. That's it. No, not done. I didn't showed only.
This this trainer will take time, huh? No, it's also very small.
They're at.
How many of you are using VS code?
I think as fast as I can and I'm downloading the data set. The advantage of collab is we can check every line and data is not saved on our laptop.
I I want to look at where the what is it doing? That advantage is that I would like you asked me.
No, no live coding. Both is not possible.

So now we are using VG validation dot text. So what we are trying to do is I'm showing how you can run a trainer just by passing a path of a file.
So earlier we saw trainer from iterators. So for iterators we pass the sequence. So here we will pass a path because the goal was to use our own data.
So can anyone tell me what this line will do? It will write it down. It has total 303 thousand text right? And then we are running for a loop.
And once we run a loop, you're passing a single single line. Now what is the syntax? So what is the syntax to write a file?
With first with then open.
Then file path, then mode. What is the mode to write W?
And there might be chances you will have encoding, so you can add that, but that is not as per the syntax right then as F.
Then what function to use for right?
F dot right? And what should be the input? It should be string. Is this clear? This is syntax.
So now we just we are repeating the same thing. What is the file path? Wiki validation text. Then we are looping through the entire validation test which has around 3000. If you want to change this as test which has 4000, we can do that.
Some of you can use test, some of you can use validation.
Someone who wants to stay here for more time, can you strain?
Is this clear? And once you do that, just check what is there inside validation if you have the data or not.
Unexpected colon. OK, I'm doing this line open.
Not while with.
OK. Thank you.
Are you able to see the file? Is it downloaded? How do we check it? We can just open this directory icon, refresh. We have the file right? OK, it should have been viki under score, but I actually wrote dot, but that's fine.
This is not syntactically correct, it should be wiki under score but.
So now what we can do is what is the next step? How do we train? What function did we use?
Earlier it was iterators right? So now that we have file, I'll just use trainer dot train.
And then I have to define a file path so that file path add it inside a list and then you have file which is wiki validation text and then trainer. Once again this should be tokenizer not trainer.

Is this clear? Is anyone train with train?
I mean, uh, the data train.
Who is playing with test here?
How much seconds did it took?
No, no. Here you have time, right? Yes. So did it take more than one second or?
OK.
So I'll unzoom the screen.
Just check this impacts.
Yeah.
Is this clear? Now what we can do is we can save it. So what is the files?
So when I open hugging phase, what files did you see? Tokenizer under score config dot Jason right? You can save that in that Jason file.
So that next time you want to use tokenizer, you don't want to train it. You can just use the saved one.
We keep the separate. Same in NLTK you save it and then save it. Same here. Also once you train any model you have to push that somewhere.
In LLM you train a model, you push it to hugging phase. Once it is saved somewhere then you can reuse it anytime. Now you can't see the difference because you're able to train in one one second, but usually LLM will take 8 hours of training, 10 hours of training. Then once it is saved somewhere then you can use it for inference.
So fine tuning of LLMS literally takes three to. There are people who train it for one day at Anubhav. I don't know if you have seen Anubhav start up at trained a fine tune model. They waited for 34 hours something. Huh.
No, something related to audio. No Excel project I'm telling Excel.
So now did it work? Did this work? OK, so now what you can do is tokenizer dot save. Tokenizer config dot Jason. So this is a standard file name, right? You can give any file name, but just make sure it is Jason.
No, now you have to run the code. Till now you shouldn't not run the code. Everything will be one. I don't know whether it is working or not.
Can you see this file? Now this is where your tokenizer is saved. Next time whenever you build anything you don't want to touch your data. Hey my data is trained, I'll just use it on new portion which user will send.
Right, so now what do we need to do? OK, my data is saved. I'll just read it from there and then use it for inference.
Now you don't even have to run all these things, so you just have to define tokenizer

read from file.
You can create a new file now. In that new file you can tell inference dot PY.
So now I'm creating inference.
Tokenizer. Now this tokenizer is a new one which is for inference.
You just have to import one line.
I'll write that line for you tokenizer.
I can equal to.
How do I use? How do I encode first using encode?
OK, I do have code below.
So here I'm using an emoji. That emoji is unknown.
So can you see this?
Now encoding gate, when you give, it will not consider any emojis. So if you see any new keyword or new token, it will be tagged as unknown. So this emoji doesn't exist. Right here if you see it's unknown and rest if you see it's working well. Now you have hello then you have comma by all is there. How is properly done. R is also properly added. You is also properly added.
And you can also print IDs just because if you want to decode back like what is the zeroth index? What is the 34 index?
Using the same logic, I'll send you a data set link. You have to train a word piece, so you need to have two files. The first file will be train dot PY, second file will be inference dot PY and you need to have one more file which is Jason.
But you will only create two file, third will be automatically created.
So basically now what you need to do for assignment is you will have train dot PY. So what will be the output for train dot PY?
What should be the output for train dot PY? It should save a Jason file and once the Jason file is saved, what should your inference dot PY look like?
Wherever the JSON is saved, read that and show the output. Is this clear for assignment what you need to do? Okay.
M.
So tomorrow we'll start with embeddings and once embeddings is done directly we'll get into RAG.
And we'll not use any libraries which are not popularly used. From now on, we'll just use libraries which are actually been used in production. So open AI one is there, hugging face, then fast embed. So fast embed is basically if you want to use any open source embeddings.

Right, you can use fast embed. What are these are the embeddings?
No, this is tokenization so far, huh? So we are currently at.
BP is done, token is done. Now we'll get into this stuff embedding layer neural network you can forget, but now embedding.
Can we use the same data set for? No, no. Again, here it was straightforward. You got to know. OK, you have to look into validation, then test. I'll give some tricky data set.
OK, which will have multiple columns.
OK.
Like doesn't matter for assignment? No, obviously, because you also need to know how to use a data set. Yeah, yeah, so I'll send the data set URL by tonight.
I do. I do.
On the.

Margi Varmora stopped transcription