Python and AI Power-Up Program Offline Class-20250924_113448-Meeting Recording

September 24, 2025, 6:04AM

1h 43m 52s

Ajay Patel started transcription

Tarun Jain 0:05

Folder source and inside this you have multiple files like search store ingest dot PY embeddings. So this is correctly done and here if you see search dot PY Ronak I guess some of the files were missing like I was not able to find state and config dot PY files.

Ronak Makwana 0:24

Actually I have restructured the code and like removed that file, but I haven't pushed the code yet.

Tarun Jain 0:31

OK, so this is not required right then?

Ronak Makwana 0:34

Yes.

Tarun Jain 0:36

OK.

So the files are correct or whatever the code implementation was there, just two feedbacks. So what you can do is you can have filtering technique since it is related to recipe right? What you can do is you can have something related to geopolitical locations.

Ronak Makwana 0:45

OK.

Right.

OK.

Tarun Jain 0:58
Let's suppose anyone ask a question like, uh, what are the best?
Help.
Recipes.

Ronak Makwana 1:10
OK.

Tarun Jain 1:11
So let's suppose I ask this question. I'm currently sitting in Bangalore, right? And here whatever data you have, it is recipe PDF. If it has something related to recipes which are relevant to Bangalore, it will give me those recommendation.

Ronak Makwana 1:16
M.
Mhm.
OK, OK, got it.

Tarun Jain 1:28
I'm just telling one example, right? Just like this, if someone is in different part of the city, if that is available in the data, you can use that geopolitical location. So this thing you can add in the UI or you can automatically take it.

Ronak Makwana 1:42
Oh.

Tarun Jain 1:45
So that is just one advancement of what you call.
Application that you will be building, so that is one on the filtering.

Ronak Makwana 1:51
OK, sure.

Tarun Jain 1:56

And 2nd what you can do is you can run evals on whatever code you have. So evals will be on 2 approaches.
So approach one is you can add MMR. Currently MMR was missing. So why MMR? Because when it comes to dishes, probably you need diversity, right? So for diversity you can use MMR.

Ronak Makwana 2:10
OK, yes.
OK.

Tarun Jain 2:21
And just take this course of it like what will be this course of MMR then approach to will be whatever existing approach. So whatever you have existing approach same thing you can use and then just get the wells.

Ishan Chavda 2:29
Yes.

Ronak Makwana 2:38
Hmm, OK.

Tarun Jain 2:38
That's it. Only this three feedback here. Remaining other things are good, yeah.

Ronak Makwana 2:41
OK, Sir.
Yeah. Thank you.

Tarun Jain 2:46
Yeah, so I also tested this friends chat bot. So here LLM providers it's loading the model right? Are they?

Hardip Patel 2:57
Hmm. Yeah.

Tarun Jain 2:59
So when you deploy this on streamlet, this is not within 1GB.

Hardip Patel 3:02
Oh.
Yeah, no, no, no. I was not facing that problem because I specifically don't intend to use the local one on on steam late.

Tarun Jain 3:14
OK, so this is first and 2nd was.

Hardip Patel 3:19
Hmm.

Tarun Jain 3:21
Where is that similarity search?
So both these things are there, right? Maximum marginal relevancy and similarity search. These are vector search only.

Hardip Patel 3:25
After.
Yeah.
Yeah.

Tarun Jain 3:35
So this is not separate sparse search, so I don't think we can have both of this. So maximum marginal relevance is kind of similarity search but with diversity.

Hardip Patel 3:42
OK.
Right.

Tarun Jain 3:50

So you can use anyone of this instead of two. And this is not sparse. This is also dense. This is also dense.

Hardip Patel 3:55
OK, OK, OK, OK, got it.

Tarun Jain 3:59
So if you want to use hybrid search with this this thing, what do you call planchain quadrant because you're not directly using quadrant client and if you want to use.

Hardip Patel 4:09
Yeah.
Yeah.

Tarun Jain 4:13
Lunch and quadrant. You have something called as retrieval mode.
Retrieval mode. If you keep it as hybrid then it will take hybrid search. So I'll show the documentation.

Hardip Patel 4:24
Mhm.
OK.

Tarun Jain 4:30
9 Chin quadrant.
So can you see my right hand side? Can you see this hybrid vectors, this keyword green color? So if you click on that you have to define fast embed and then define quadrant BM25. This we have already done.

Hardip Patel 4:40
Yes, yeah.
OK.

Tarun Jain 4:53
When you are creating the collection, this is also same. If you notice you have vectors

conflict dense, sparse conflict sparse. So till here everything is same. Now what you need to do is when you define your quadrant vector store you have to define your retrieval mode.

Hardip Patel 4:57
Hmm.
Right.

Tarun Jain 5:12
So earlier what did we do? When we define quadrant vector store, we define client collection embedding. These are the only three things that we define.

Hardip Patel 5:14
OK.

Tarun Jain 5:22
But if you need hybrid search, you have to add sparse embeddings. Then you need to add retrieval mode. So what is the retrieval mode? It is hybrid.

Hardip Patel 5:22
Yeah.
OK.

Tarun Jain 5:35
So this particular application doesn't have hybrid search yet. Even if you use BM25, it's not been used.

Hardip Patel 5:38
OK, so uh, just a question.
Yeah.

Tarun Jain 5:45
Are you got it here?

Hardip Patel 5:47

Yeah, yeah, yeah. Only thing is, OK, so this is one question. Let's say if I try, I want to use hybrid search right now, I'll need to do the embeddings again, right? I mean, I need to load the vectors again, right?

Tarun Jain 6:01
Ingest, correct? Yeah, so where is that code? I was not able to find the ingest code.

Hardip Patel 6:03
OK.
Oh, ingestion code. I did it separately on Colab only. I'll add it. I'll add it tomorrow.

Tarun Jain 6:14
OK, because when you look at this quadrant vector store it they it doesn't have hybrid search thing. If you use quadrant client directly using points, which here it is there this one.

Hardip Patel 6:19
Mhm.
OK.
Hmm.
Hmm.

Tarun Jain 6:31
This is using hybrid search because here you have prefetch.

Hardip Patel 6:32
Right. Yeah, yeah.
Yeah.

Tarun Jain 6:40
Yeah, uh, mainly that's it.

Hardip Patel 6:43
Yeah, uh, understood. Thank you.

Tarun Jain 6:46
OK, so let's get started.
So what you can do is you can create a new project folder and you can name it as evals or anything.
Uh, let's write it on VS code itself.
And meanwhile you have this CSV file, right, the one which. So what we'll do is first let's open Colab and try with this raga's data set. So why am I sharing this data set here? So let's suppose whenever you run evals, right?

Hardip Patel 7:16
What?
Yes.
OK.

Tarun Jain 7:33
There might be high chances you will create your own evals. So what did we do here yesterday? Yesterday we generated the data set.

Hardip Patel 7:38
Good.

Ronak Makwana 7:42
Yes.

Tarun Jain 7:43
So if you see a data set creation, you define your Open AI embeddings, then Open AI LLM and then Persona. Once you have Persona, you're generating the data set. So whenever you have the data set in this particular format, what is this library called? If it is in uh tabular column.

Ajay Patel 8:09
Pipe and no, not Pipanda.

Tarun Jain 8:10

Pandas right? So let's suppose Pandas is nothing but your CSV file. Now just imagine you created your own data set on Excel sheet where you have your own questions, you have your own reference context and as well as responses. So if I open this.

Ronak Makwana 8:10

Pandas.

Ajay Patel 8:16

Mm.

Tarun Jain 8:28

Raga dot data set which has been uploaded now. So what we can do is we can create a new collab notebook.

So this created.

So what you can do is you can make the subbedding as load your own data.

Or if you have already generated the data set, let's suppose you already generated data set from Ragas and you want to save it so that future if you want to run the evals again, what you can do is you can save that as a CSV file. So how do we save that?

You just have to run data set to pandas. Then there is a function called. I'll just copy this, save it here. Then you have something called as to CSV. Give a file name. I'll give Raga's data set.

dot CSV that's it. And then what you can do is you can keep index to be false. This will download that particular file and I will show you the syntax.

So did you create a new collab? Load your own or load the generated CSV data?

Hardip Patel 9:47

Oh.

Tarun Jain 9:55

So now what you can do is you can upload that CSV file.

Hardip Patel 9:56

Alright.

Sorry, do should we do it with VS code or just the colab?

Tarun Jain 10:06
No, first we'll just check the format how this particular data set looks like. Then we'll jump to Pallab. So if you look at this code, I've just loaded the data set. So what does this data set of user data reference, context reference. So this is added by domain expert which are actual data.

Hardip Patel 10:07
It.
Welcome.
OK, OK.
Mhm.
Oh.

Tarun Jain 10:25
Now what we need to do is we need to add retrieved context and responses from LLM. Once you have this thing then we will go to collab. I mean VS code.

Hardip Patel 10:31
Oh.
OK.

Tarun Jain 10:36
To generate the retrieved context and response.

Hardip Patel 10:41
OK.
OK, I read these CSV.

Tarun Jain 10:45
Import pandas as TD.
And then just define the path. So path you can copy from here.
And then what are you supposed to do? Just define your data equals to PD dot read CSV.
So if I want to print top three columns, what will be the top three rows? What is the

comment? Anyone remembers?
For top three rows.

Hardip Patel 11:23
Oh.
Uh.
Yes.
Data square versus column three dot oh sorry dot get column three.

Tarun Jain 11:40
No Head of 3.

11:42
Head.

RamKrishna Bhatt 11:45
B.

Hardip Patel 11:45
Oh, OK Oh.

Tarun Jain 11:49
So if I want bottom 3.
Data dot.

RamKrishna Bhatt 11:54
And.

Tarun Jain 11:55
Tail 3.

Hardip Patel 11:55
Minus.
The.

Tarun Jain 12:03

So this is the data we have user input, we have reference context, we have reference. As of now, whatever data we have, it is in CSV format. But what are we supposed to do? We need to convert the CSV.

Convert this CSV into Ragas evaluation data set.

So yesterday you saw this command, right? Where is it?

Can you see this line eval data set equals to whatever data set you have, you're converting that to evaluation data set. So yesterday we used this approach. Today what we are trying to do is we already have the data which is in CSV. We want to convert the CSV into this particular data type.

Hardip Patel 12:44

Oh.

OK.

Tarun Jain 12:55

This eval data set. So what is the data type of eval data set? Can you see this evaluation data set? Then you have feature, user input, retrieve context, reference context and response. So we need it in this format.

So let's install ragas.

You understood what we are trying to do.

Hardip Patel 13:23

Yeah.

Tarun Jain 13:24

So whatever data you have, let's suppose you have your own CSE file. You're trying to convert that into evaluation data set format. That's it.

Let load evils to robbers.

OK, so.

In Singleton yesterday we had five variables, correct? We had user input.

We had reference context.

We had reference, so these are actual information.

Then you had retrieved context.
And then you had response. So what is the data type of user input?

Hardip Patel 14:21
String.

Tarun Jain 14:21
String reference context.

Hardip Patel 14:24
Play Stop string.

Tarun Jain 14:26
List reference.
Reference is nothing but the actual output.

Hardip Patel 14:32
Yes, string.

Tarun Jain 14:34
Then then retrieve context. After search you'll get list and this is the LLM response which is string.

Hardip Patel 14:37
List of.

RamKrishna Bhatt 14:38
List.

Ronak Makwana 14:43
Thing.

Tarun Jain 14:45
So this is how your data type needs to be like. Now what you can do is can you check the data type of reference context? So just do data of reference context.

You will get this particular what you call. You'll get this particular column. Then just do I lock which is nothing but your index of 0.
You will get the first index value. Now if you check the type of this.
It is in string, but what data type do we need?
We need it in a list. So what you can do is you can just run one single line data of reference context.
Equals to.
So whatever auto complete you have, it's correct.
Eval is nothing, but it will convert that into a responsible what you call. Let's suppose you have string. Inside this string you have a dictionary name equals to.
Tarun.
So what is the specific data type of this? What is the data type?

Hardip Patel 16:09
Uh, stop doing.

TJ

Tarun Jain 16:09
If I check the type of this.
It is string, but technically what this should be?

Ronak Makwana 16:13
And.

Hardip Patel 16:16
Jason, uh, this should be a a dictionary.

TJ

Tarun Jain 16:18
Dictionary right? So what we will do is you will use an eval function and then if you check now it is dictionary.

Ajay Patel 16:19
List.

TJ

Tarun Jain 16:27
You got it. So eval is what it is trying to do is whatever is within the string, what data

type it is there, it will convert into that.
So what is this called? This is a function and we also saw this command in pandas. If you want to make any changes in your column you have to use a function called apply. Inside apply what are you supposed to add? You need to add function which is eval.
And one example what we saw is we saw stop words which is def remove stop words. And then you have some logic. Now what will you do? Whatever data text you have, you want to remove the stop stop words from it. We just do data dot TXT dot apply then remove stop words.

Hardip Patel 17:24
Um.

Tarun Jain 17:24
This is stop words.
So you know this syntax, right? What apply will do? Apply will apply certain transformation for that particular column, and the input that it will take is a function. That function needs to be a logic, so that logic can be removed upwards, it can be lemmatization, it can be stemming, it can be anything.

Hardip Patel 17:45
OK.

Tarun Jain 17:46
In our case, what is that function name? It is eval. Is this clear?

Hardip Patel 17:48
Mhm, mhm.

Tarun Jain 17:51
So now if I just run this and now if I run the same line here.
Type of data reference context I log zero. Now it should be list.
Is this clear now? So this column is supposed to be in list. It is in yeah.

Hardip Patel 18:04
Hmm.

Ajay Patel 18:05
It's giving me error data of reference context dot apply eval but I guess it should be a string or.
Events should be a.

Tarun Jain 18:17
This one.
No eval should be a function.

Ajay Patel 18:21
Events should be, but we don't need to define now. I I I guess it's in build.

Tarun Jain 18:23
So if you.
That is inbuilt. So if you see a type of eval and what am I doing? I'm adding this parenthesis inside parenthesis I have checked. So what is this called? This is a function.

Hardip Patel 18:33
Yeah.

Ajay Patel 18:39
OK.

Tarun Jain 18:39
So if I just run eval, this is an inbuilt function.

Ajay Patel 18:43
Hmm.

Tarun Jain 18:44
So that is what we need to add in apply. Apply takes function as an input.

Ajay Patel 18:48
OK, but when I try to run this it gives me an error.
I can type error.

Hardip Patel 18:54
Oh.

Tarun Jain 18:54
So I hope you didn't add like this.

Ajay Patel 18:59
Evil argument one must be a string, bytes or code object.

Hardip Patel 19:05
Uh.

Tarun Jain 19:07
OK, can you check the spelling if this is correct?

Hardip Patel 19:11
Yeah.

Ajay Patel 19:11
Just a second. Yeah, that might be a chance. Reference contest.

Tarun Jain 19:19
How many of you are getting errors?

Ajay Patel 19:23
No, spelling is fine.

Tarun Jain 19:28
Oh, can you share your screen?

Ajay Patel 19:29
Yeah, sure.

Hardip Patel 19:33
But.

Ajay Patel 19:34
Sharing the screen. Can you see?

Tarun Jain 19:37
Oh, yeah.
One second.
No, why is it Lambda?

Ajay Patel 19:45
No, actually that's an auto suggestion.

Hardip Patel 19:46
Just click on cancel like this in the bottom in the bottom.

Ajay Patel 19:52
Yes.
I.

Hardip Patel 19:58
Except run, accept and cancel.

Ajay Patel 20:05
Hmm.

Tarun Jain 20:06
Cancel one second.


Ajay Patel 20:07
Apply e-mail OK data context reference context dot apply equal to.

Tarun Jain 20:13
Oh, can you just print a data reference context once?


Hardip Patel 20:20
Oh.


Ajay Patel 20:28
That's a column.

Tarun Jain 20:30
OK uh so can you print this thing? I have sent one line.
OK.
Can you print this one?


Hardip Patel 20:41
Oh.

Tarun Jain 20:45
OK, and in your it's already list.


Ajay Patel 20:45
List.


Hardip Patel 20:47
Oh yeah.


Ajay Patel 20:47
OK, so I don't need to do this now.

Tarun Jain 20:51
Yeah, only you don't know too. I guess only that. So only you got a list. So now can you see my screen again?

Hardip Patel 20:51
So uh.

Ajay Patel 20:58
OK, I'll stop sharing.

Hardip Patel 21:02
Might have been run.

Tarun Jain 21:02
So let's suppose I'm here. So if you see here I loaded my data and once I loaded the data, if I check the type of it, it is string. Then what I'm trying to do, I'm changing the type version. Now it is list. Now if I run this line again, it is an error.

Ajay Patel 21:08
Hmm.
OK, OK.

Hardip Patel 21:12
Hmm.

Tarun Jain 21:20
This is the same error that you have got.

Ajay Patel 21:24
Makes sense.

Tarun Jain 21:25
So I'll just click on this, I'll comment this out. So I hope everyone just check this if it is in list or not. This is what we need.

Ajay Patel 21:33
Hmm.

Tarun Jain 21:34
And now what we can do is from Ragas we can import evaluation data set.
Why is it so slow evaluation data set?
And then you have eval data set.
Equals to.
Evaluation data set dot Can you see this function from pandas?
We just have to select from pandas and add your data.

Hardip Patel 22:14
Um, yes.

Tarun Jain 22:21
That's it. Now if you print this eval data set, it should show like this feature features, user input, reference, context and reference, same as what we saw yesterday.
We need this format so that you can pass it inside evaluate. So this function over here the evaluate we will run it on VS code.

Hardip Patel 22:46
OK.

Tarun Jain 22:55
So now the question is how many of you have your vector index? Since we are using the same code, whatever index you have created inside your quadrant vector store, how many of you are using the same thing which is related to Atyantik?

Hardip Patel 23:11
Me.

Ajay Patel 23:13
Hmm.

Tarun Jain 23:14
I mean, is there anyone who is using some other data set?
Instead of the data that we had.
Like is anyone using the PDF for any other data or is it Atyantik website itself?
Anyone is using different? No, right?

Hardip Patel 23:36
I am I'm using same but the collection name and everything is different.

Tarun Jain 23:40
Collection name is fine. So what I'm trying to say is so when we run this evals, this evals is related to the authentic chatbot. So if you have something else you might face some different results, you might get 000.

Hardip Patel 23:44
Yes.
Um.
Oh.

Tarun Jain 23:56
So that's the reason I was asking. If you're not, I could have shared my quadrant API key and URL.

Hardip Patel 23:58
No.
Mhm.

Tarun Jain 24:04
OK, so now what we can do is we can come to VS Code. Now the goal is whatever we did in Collab, we will dump that in VS Code and we'll try to understand how to structure the project files, right? And then.
Since we uploaded our own CSV data set after this, what are we supposed to do? We will just repeat it.
So whatever we did yesterday right to get this result.

We will just replicate it because yesterday you didn't write code, I was just running it. So today we'll write it along with me. That's it. So no new code will be written. It's just we are repeating. That's it.
But in a proper modular way.

Hardip Patel 24:53
OK.

Tarun Jain 24:54
So what we can do is what kind of libraries do we need?
We need Ragas, then we need fast embed for embeddings, then quadrant client, Langraf, Langchain and Langchain Google Geni. So I will just copy this, paste it here.
So make sure we remove this exclamometry mark.
So the first step is we create environment variable. Second step is we have to run the installation step.
So Python 3 we when we.
PNV. Then what is the next line? We need to activate source PPNV.

Hardip Patel 25:42
Been activated.

Tarun Jain 25:42
Bin then activate and once this is done just run this pip install.
Let me know once this is done, then we'll start.

Hardip Patel 26:27
It.
OK.

Tarun Jain 26:58
To those who have already completed, what you can do is then you can create an ENV file just dot ENV and once you create dot ENV we need to add Google API key, quadrant API key, quadrant URL.

Hardip Patel 27:08
Yeah.

Tarun Jain 27:12
And if you want to use OPIC, you can also add OPIC API key.

Hardip Patel 27:17
Um.

Tarun Jain 27:22
So this should be inside dot TNV.

Hardip Patel 27:40
OK.
You are it.
OK.

Tarun Jain 27:46
Project structure plan dot CY dot PY.
Models dot VY or C model dot VY.

Hardip Patel 28:13
So we use our own API key, right?

Tarun Jain 28:16
Ha ha.
Google quadrant quadrant URL opic is optional.

Hardip Patel 28:27
Done.

Tarun Jain 28:30
Then state dot PY you have to use dot PY. OK, so now what we can do is if in case you have completed, just create all these files inside your project folder. One is client

dot PY. I'll tell you why we need all this. One is config dot PY.
And then main dot PY, models dot PY or schema dot PY. So you can pick any one of it. Either you can have models dot PY or you can have schema dot PY and then state dot PY and then utils dot PY.
And we can also have evals dot PY.
Creative vals dot PY.
So total how many files 12345671234567 and then you will have this dot ENV dot ENV is not a file so I'm not counting this.
Totally you should have seven. This is optional. We don't have database, so as I said we don't have database. In most of the cases you'll also have database dot PY. If in case you want to save login details of the user or any other details then you'll use data database dot PY. But we don't have this.

Hardip Patel 30:10
Oh.

TJ

Tarun Jain 30:14
It can be empty files. You can create empty files.

Hardip Patel 30:17
Long term start.

TJ

Tarun Jain 30:21
Then just count if you have 7 files.

Hardip Patel 30:35
Yeah.
1.

TJ

Tarun Jain 30:48
Oh, is it done?

Hardip Patel 30:52
Uh, yeah.

Tarun Jain 30:55
OK, everyone, right?


Hardip Patel 30:57
Uh, I don't know.

Tarun Jain 30:59
Guys, can you confirm if you have completed?


Ayush Makwana 31:02
Yeah.


Mitesh Rathod 31:03
Just a second. Can you please show the sidelines? Yeah.

Tarun Jain 31:07
Yeah.
So basically what client dot PY will have is let's suppose you define any LLM right LLM client. You can define LLM client. You can define your embedding here. So let's suppose what is the function that we use for the. Let's suppose I have.


Hardip Patel 31:32
Um.

Tarun Jain 31:41
Plan chain and have LLM. If I want to ask a user input and if I want to execute it, what function are we using to run this LLM LLM dot what are we using?


Mitesh Rathod 31:52
Invoke.

Tarun Jain 31:54
We're using invoke, right? So this is in line chain.

Now for open AI, how did we run it? You have client, then you had create chat completion.

Hardip Patel 32:09
Oh.

Tarun Jain 32:09
This is how it was defined in open AI and we also saw hugging face. In hugging face we had client dot we had text completion.

Hardip Patel 32:20
Mm.

Tarun Jain 32:21
This is for rugging face. So if you see this is nothing but a class. Inside that class you have a function called invoke. So this is what we will do in your program as well. If you want to define any custom function to execute the LLM, you can use it.
Why? Because whenever you define LLM, it's not that you are using it in one file itself. So now just take a look at this files. Where do you think LLMS will be used?

Hardip Patel 32:51
I.

Tarun Jain 32:53
So you have client dot PY inside which you'll use an LLM. So what I'm saying is LLM is not used in just one file. It will definitely be used in multiple files and you want to maintain a structure. Since we want to maintain that structure, we are creating a new file and we are maintaining that structure.

Hardip Patel 33:01
Mhm.

Tarun Jain 33:12
Now that same thing will be repeated in multiple files. So in evals we need LLM. Here we need LLM.

Hardip Patel 33:20
Oh.

Tarun Jain 33:21
Then there might be chance in main dot PY we need LLM.

Hardip Patel 33:25
Oh.

Tarun Jain 33:26
And then in state dot PY we need LLM.
Why? Because you have a node answer node we have inside this answer node you're using LLM.
So is this clear? Since we have it in multiple files, we have to maintain that structure.
Now coming to config dot PY, what we are supposed to do is whatever API keys you have, you have to define that in the config dot PY. For example, let's suppose.

Hardip Patel 33:38
Hmm.
Yes.

Ajay Patel 33:44
Yes.

Tarun Jain 33:55
Tomorrow Gemini 2.5 Flash is deprecated. What are you supposed to do? You will search in all the files where you are using LLM. You have to change it from everywhere. So in order to avoid that issue, let's suppose here you have LLM.
In Evals you have LLM and in state you have LLM. Everywhere you have Gemini 2.5 Flash.
So now if you want to remove this from every single places, obviously we can do a global search and you can remove. But if you want to avoid that and avoid the reputation, we can have something called as config dot PY which will have all the details. So now if some other day Gemini 2.5 flash is replicated, I will just.

Change it here. I'll make you this as the pro.
So understood. So in config dot PY we just have to define all the variables. OK, why is Azure coming here? I'll remove Azure. We don't have Azure. So model name is there, temperature is there, maximum tokens. If you want to edit right, you don't have to search in other files.

Hardip Patel 34:45
Home.

Tarun Jain 35:02
You only have to come to one single file and make the changes.
So this is nothing but a pydentic class and this class has low DNV. So for this low DNV how many variables do we have?
We have Google API key, quadrant, quadrant, URL and OPIC. So if you want to define your collection name here, you can define your collection name here. Collection name equals to hybrid.

Hardip Patel 35:18
4.

Ajay Patel 35:19
Total 4.

Hardip Patel 35:26
Mhm.

Tarun Jain 35:31
And if you want to define your embedding models, you can define that here. So you have both the options, but I prefer using it in config dot PY. So you have model name, model name also we don't need in environment, but if you want to use environment it's fine. If it is not present in environment it will use it.

Hardip Patel 35:33
OK.
Yeah.

Tarun Jain 35:50
From this default same goes for temperature. Here if you have not defined temperature it will pick 0.0. So if you want to change it you just have to change this value.

Hardip Patel 35:52
OK, yeah.

Ajay Patel 35:52
Mm.

Hardip Patel 35:55
Mm.
Mhm.
One more question Tarun. I have used Azure and for five GPT five it shows that that you no longer need temperature.

Tarun Jain 36:03
Can you?

Hardip Patel 36:18
Yeah, GPT5. Do you know any?

Tarun Jain 36:19
For which one GPT 5?
No, I all those parameters are configurable because there will be times. OK, that is a reasoning model, right? I'm not sure.

Hardip Patel 36:29
Oh.
It is reasoning model as well.

Tarun Jain 36:35
It.

Hardip Patel 36:44
Yeah, I see.

Tarun Jain 36:52
OK, maybe it's not want on it. Temperature top are not needed in GPT 5 models.

Hardip Patel 36:54
OK, maybe.
Yeah, I.

Tarun Jain 37:05
OK. When did they remove August 12th last month only?

Hardip Patel 37:09
So like if you run GPT5 it is giving error if we add temperature.

Tarun Jain 37:15
However, we are not using GPT 5. It's costly and it's not that great compared to GPT 414.1. It's working fine, but yeah, I was not aware of this.

Hardip Patel 37:22
Yeah.
Yeah, yeah.
Yeah, even like a mini or nano version of this is cheaper, but even there it does not work temperature.

Tarun Jain 37:35
Temperature.

Hardip Patel 37:38
Yeah.

Tarun Jain 37:40
OK, I I was not sure about that.

Hardip Patel 37:42
Sorry to go on a tangent.

Tarun Jain 37:45
Yes, well usually you you define that here itself, so all the parameters whatever you have right. So since you are repeating LLM in multiple files, we have to define that structure. So even if it's not supported, you can define that here. So if there is any LLM which requires temperature then you can just load it.

Hardip Patel 37:59
Mhm.
Mhm.

Tarun Jain 38:07
So this is for LLM. Then you have quadrant. So if you see quadrant API key, I'm getting it from environment which is.
Quadrant API key. If it is not defined you can keep it as none. Same goes for quadrant URL. I'm getting it from ENV which is quadrant URL collection name, then open AI API key. I was using this for evals and then you have dense embeddings which is Xena sparse embeddings quadrant.
And then just define this class. Once you define this class, you can just instantiate it. I'll copy this file. You can paste this in config dot py.
And I hope you understood the syntax. This is the same pydantic. We are just defining the variables data type and we are assigning a value. This value we are reading it from environment variable. Some of it it is directly defined like dense and sparse embeddings.
So we have 123.
Wait, one more. We forgot. We forgot.

Ajay Patel 39:23
Open APA.

Tarun Jain 39:25
Google API No Open AI is there. Oh right, Open AI. It's not there here.

This I will paste and Google.
APIE.
STR equals to OS dot get in only Google.
You can add this one also.

Hardip Patel 39:59
So now what?

Tarun Jain 39:59
So now what we can do is you can have client dot PY just from config.
Import uh.
We can just do from conflict.
Import.
This variable config.
Now if I do print config.
Configure dot.
Model name. Then I will also print.
Configure dot.
Temperature.
Just check if it is printing or not.
I'll do Python source.
The PNV activate Python client dot PY I have Jimmy 92.5 pro and 0.0. Just cross check if you are able to print this.
So understood in client dot PY we'll define our LLM and embedding config dot PY. Whatever input variables you have which is repeated in multiple files. What you can do is you can define that inside configure evals. I guess it is straightforward.

Hardip Patel 41:33
Mm.

Tarun Jain 41:36
If in case you want to directly run ragas, we will be using evals. So here is where you will have your data set.
Right main dot PY will be your main logic where user will add the user input. So user input will be added here.

And if in case you have any Screamlit app, that Screamlit app will be here. So this is the universal file where the actual user input will happen. Then you have models dot PY. So let's suppose you have output parsers, right? So for output parsers, what do we do? We define.

Hardip Patel 41:57
Yeah.
OK.

Tarun Jain 42:13
And we have multiple schema right? So if you check the readme file, I wrote it as models dot PY or schema dot PY. So whenever you're using output parsers, all the class that you have right, it will be here.

Hardip Patel 42:14
Mhm.

Ajay Patel 42:22
Hmm.

Tarun Jain 42:29
I'll show one example.

Hardip Patel 42:30
Mhm.

Tarun Jain 42:31
Let me copy from one of the.
Other code that I have.
So if you see you have user data and phone number, name, user input, off topic questions, conversation completed and then you have API key, then you have tokens.
So whatever schema you have to define it, that schema will come inside.
Models dot PY or schema dot PY if you are using output parsers.
Is this clear what models dot PY will have? In our case we don't have models dot PY.
Also we don't have.

Output parsers. So this will be empty in our case, but whenever you're using output parsers, make sure you're defining that inside models dot PY. And one best example is you have this answer relevance, right? Yesterday we had answer relevance.
And context relevance.
So let's suppose you want to print the Jason. What are the 2 fields you need here?

Hardip Patel 43:59
Um, let's see.
Um.
Just.

TJ **Tarun Jain** 44:07
One is output and one more is the justification.

Hardip Patel 44:12
Good.

TJ **Tarun Jain** 44:13
So this is boolean.
And this is string.
So if you want to define the class schema for it, you'll be using models dot PY and then you have state dot PY. So state dot PY will have search.
And it will have answer which is your land graph code.

Hardip Patel 44:37
No.

TJ **Tarun Jain** 44:37
And then you utilities dot PY so if you have any error handling functions.
Error handling functions like let's suppose when you are using this output parser you were not able to convert it into Jason. Then what are the error handling techniques you can use? So wherever you have type try and accept block rate.
Wherever you have error handling messages that will be added in utils dot PY. Is this where the project structure?

Hardip Patel 45:09
Yes.

Tarun Jain 45:09
So I hope this is completed config dot PY and you're able to get this results.

Hardip Patel 45:10
Cool.
Yes.

Tarun Jain 45:16
So till here is it done, you got the results. Anyone is getting error?
Anyone is getting error or should we proceed?

Hardip Patel 45:34
Um, I'm fine.

Ajay Patel 45:36
We should proceed. We should proceed.

Mitesh Rathod 45:37
Yeah, we can proceed, yeah.

Tarun Jain 45:38
OK.
So here what we'll do is let's just copy this line, the Google generative AI one and paste it here and let's just define LLM client.
Deaf in it comes first, then self.
And here I'll define self LLM equals to copy this chat Google generative AI and here you can define model.
Do we have model here on figure dot model name?

Hardip Patel 46:12
Yes.

Tarun Jain 46:18

Then we have temperature.
Configured dot temperature.
And then we have max tokens.
Voice to configure dot Max opens.
And here what I'll do is now every time I use LLM I will have my own function instead of invoke. What will I do is I will define get response is good function. I'll use get response on the get response.
Self. Now I hope everyone knows why I'm using self because this is a class method.

Ajay Patel 46:52

Mm.

Tarun Jain 46:55

Self then comma prompt which is string.
Response equals to. This is the attribute. Here I will use invoke.
Prompt and then I will return response dot content.
So let me copy this.
Paste it here so this should come inside to client dot PY.
And if you want to test it, let's just define.
Meta class.
The main and I will define the client equals to.
LLM client.
Now the prompt is who is the main character of one piece.
And then you can just do print client dot get response prompt.
Now if I print Python of clarin dot py.
So if you see the main character of one piece is Monkey D Luffy and then you have the response, he's the captain of slot pirates, blah blah blah.

Hardip Patel 48:26

Hmm.

Tarun Jain 48:26

Just cross check if you got the output.

Hardip Patel 48:31
Yeah.

Ajay Patel 48:32
Yeah.

Tarun Jain 48:36
OK, so the next thing is always keep this in comments. This is only for testing purpose.
This is only for testing purpose.
Or.
Individual files.
I hope this syntax is clear, right? We define a class, then we have init function. Inside init we have a class attribute which is self LLM. This self LLM is repeated when I'm doing get response.

Hardip Patel 49:16
Mhm.

Tarun Jain 49:18
Now what is the next thing we have to define our class?
Embedding client. So how many class attributes will we have here?

Hardip Patel 49:24
OK.
Mhm.
I will.

Tarun Jain 49:37
Uh, what class had? Here you had one here. How many will you have?

Ajay Patel 49:44
Class attributes.

Hardip Patel 49:45
Uh, this will be uh uh, I guess dense. Uh.

Tarun Jain 49:51
Correct. One is dense and one more.

Hardip Patel 49:53
What?
And then for this bars.

Ajay Patel 49:55
Sparse.
Hmm.

Tarun Jain 50:00
So let's import this line, same whatever. There is no new code we are writing. The only thing is we are putting that inside class and we are keeping it well maintained.

Hardip Patel 50:11
Oh.

Tarun Jain 50:11
So we had we need to add this in import statements.
And then embedding client def init self self dot I will define dense model equals to copy this text embedding.
And you have model name equals to configure dot.
Uh, dense embedding model. So next time you want to change any embedding model, just come here, change it. You don't have to see where you have used that.

Ajay Patel 50:44
Mhm.

Hardip Patel 50:44
Oh.

Tarun Jain 50:49
So self dot sparse model sparse embedding model. I hope this spelling is correct.

Hardip Patel 50:58
Yes.

Tarun Jain 50:59
Oh.
So this will be direct. So self dot dense model text embedding model name equals to configure dense embedding model. Then same goes for self dot sparse model equals to sparse text embedding and this is beyond 25.
And where are we using this? Where are we using this particular function?
Sparse embedding. I'll just copy this.
We are using it at three places. One is this.
We are using it in the state. If you notice this, this is next and if we are using next, what is this called? This is called iterator.

Hardip Patel 51:43
Hmm.
Mm.

Tarun Jain 51:51
So whenever we return next, we need to use a function called yield.
So so far whatever we are doing is we are using return but whenever we are using next and if you have iteratable variable we have to use yield instead of return.
So what I will do is.
Uh, for this what function can we give? We can either define query embed or we can have something meaningful. So def.
What we can do? Query embed. Then I'll define for dense self comma. So what is the input here? It is query. Query is string.
Then you can copy the same line. Copy the same line.
Paste it here. Now this self dense embedding model will be replaced with what?

Hardip Patel 52:56
Uh self dot uh dense model.

Tarun Jain 52:57
dot dense model and this will be replaced with.

Hardip Patel 53:03
Query.

Tarun Jain 53:04
Very.
And let's just see what autocomplete will give. It should give yield. It's giving return which is wrong. They should be yield.

Hardip Patel 53:17
Mm.

Tarun Jain 53:23
And then I'll do autocomplete query embed sparse self. Then you have query. Then you have STR self sparse embed query, embed query, query embed query and then next. Next is there whenever I'm using next by default I'll use yield.

Hardip Patel 53:38
Yeah.

Tarun Jain 53:46
So where should we test this now? We will test this in state dot PY directly. We will test it here.
So let me copy this and.
Add it here. So far everyone understood it what we are trying to do.

Hardip Patel 54:00
D the D

Ajay Patel 54:03
Hmm.

Hardip Patel 54:05
Yes.

Tarun Jain 54:06
We are just defining the client. Once we define client, we need to test it. So here I can test it directly, but The thing is it's better if it has this directly on state dot PY. If not, let's just test it.

Hardip Patel 54:18
Yeah.
OK.

Tarun Jain 54:24
So this will be embedding client.

Hardip Patel 54:25
So like we we are like we already use like embedding client classes or to import and all right. So why do we need to comment this? What if we just keep it?
And comment that, yeah.

Tarun Jain 54:41
So if you run this right, you're using this what you call client dot PY.

Hardip Patel 54:47
Hmm.

Tarun Jain 54:47
What you call here you will import it correct. So from client import you will do LLM client comma embedding client.

Hardip Patel 54:50

Yeah.
Yeah.

Tarun Jain 55:01
So every time you run this or what you call, every time you run this particular line, this will be executed.

Hardip Patel 55:06
Huh.
Oh, is that?

Tarun Jain 55:10
Huh.

Hardip Patel 55:11
Oh, that's right. OK.

Tarun Jain 55:13
So usually if you see most of the time, whenever you test it right, you test it and you comment this. In most of the cases you won't comment, you will remove this off.

Hardip Patel 55:17
Yeah.
OK, OK.

Tarun Jain 55:23
So you can check in most of the open source library. Uh, open ACI.

Hardip Patel 55:25
Uh.

Tarun Jain 55:31
So let's suppose I want to test clot dot PY. So I define clot dot PY inside this app run function and inside this run if you see you have the LLM self dot LLM. So what will I do? I just have to do.

Hardip Patel 55:34
OK.
So.
Mm.

Tarun Jain 55:48
From open AGI import chat entropic model then client dot run. So if I want to test this separately I will have a main function which I don't have here because this is used in different files.

Hardip Patel 55:51
Right.
Mhm.
Mhm.
Right. OK, got it.

Tarun Jain 56:08
This is only for individual file testing.
So here I'll do client dot query embeddings and you can ask any name. I'll do hello world.
So this will print vectors.
Python client dot PY.
So if you see this is a generator. Now here if you want to see actual output, it is better if you use it in state dot PY because when you run state dot PY you will get some clarity what was returned, what was the relevant documents for the given user query.

Hardip Patel 56:48
Oh.

Tarun Jain 56:56
But this is working. If you see I got the output.
So before I close this file, just cross check if you have the same things from config import config so that we can use model name, temperature, Max tokens, then LLM.

Why? Because I need to define Google generative AI and for embedding I need text embedding and sparse embeddings.
So you have LLM component in it. Inside in it you have LLM, then get response, then class embedding client. You have query embed dense, then query dense parse.
And this one I will remove it.
Is this done earlier?

Hardip Patel 57:48
Yes.

Ayush Makwana 57:49
Yes.

57:50
Yes.

TJ

Tarun Jain 57:51
OK, so client dot PY is done. I will close this.
Config dot PY is done. I will close this. UTS dot PY will come later. Read me will come later. Evals later. Models we don't have main later. Now we'll focus on state dot PY.
So the first line for state dot PY is from client import LLM client and embedding client.
Then what will I do is I will come back to our code. Whatever search function I have, I'll copy the entire thing. I will paste it here and for time for this purpose instead of state, let's define query.
Once we'll test, after we test, we'll convert it into a state.
So what this line will be now?
So this line I will remove both these lines.
And 1st what I will do is I'll define the embedding client.
Embedding client equals to embedding client. Then you have dense vectors. Dense vectors equals to embedding client dot get dense vectors. This is the function that I've used.

Hardip Patel 59:06

Yeah.
OK.

Tarun Jain 59:20
If you see we have query embeddance, query embeddance and then user query, user query. Copy this.
Come back here and defines parts vectors.
Equals to embedding client dot query for some query and then you can add next again.

Hardip Patel 59:51
Um.

Tarun Jain 59:57
OK, this is sorted. Now how do we import these models?
From quadrant client import models.
And we need collection name. So where do we have the collection name?

Hardip Patel 1:00:15
Uh, configure.

Tarun Jain 1:00:17
config, so from config import.
Configure. I will copy this.
And then you have collection name. What else is pending? This client. This client is from quadrant so.
Here I have to define quadrant client.
And.
Client equals to.
Modern client.
I have URL. URL is nothing but configure. OK the auto complete is correct. One is URL and one more is API key if I can check.

Hardip Patel 1:01:01

Oh.
Yes.

Tarun Jain 1:01:06
URL and API key and this is coming from configure. That's it.
So let me test this out if.
Name it should be made it to contact at the and let me check you.
The collection name where is collection name hybrid and in my VS code also it is hybrid, yeah.

Hardip Patel 1:01:41
OK.

Tarun Jain 1:01:42
So now if I print Python of state.py, So what should it return?
It should return context. Inside that context I should have a list.
So it returned me the output. So I have context inside context I have list and this will be length of three.
So I didn't change the code. The only thing what I did was I changed this particular variables. That's it.
Add this inside state dot PY.

Hardip Patel 1:02:30
Um.

Tarun Jain 1:02:32
And what is the other node we have?
What is the second one def?

Hardip Patel 1:02:40
Answer.

Tarun Jain 1:02:42
Answer generation. So this will also take query and it should also take context.
As of now we are testing it, but once we test it then we will change it into state.

So this is the answer generation. How many input it has? One is context and one more is state query.
I will copy this answer and I will remove this rack state.
It has join which has state context. So where is the state context coming from? So I'll just add query. Then here what I will do is I will call the search function documents equals to search of query.

Hardip Patel 1:03:24
Um.

Tarun Jain 1:03:33
Then this search.
Can anyone tell me what should I write here inside join?

Hardip Patel 1:03:39
OK.
Just a second. Inside. Sorry. Oh.

Tarun Jain 1:03:47
So here you see everyone understood this logic, right? Search logic. I want to use search. So for search I need to give the user query. So I will give that user query.

Hardip Patel 1:03:50
Yeah.
Mm.
Yeah, we got the dogs.

Tarun Jain 1:03:59
You got the docs. Now what should I write here inside join?

Hardip Patel 1:04:04
Docs dot page content.

Tarun Jain 1:04:09
It will it be page content?

Hardip Patel 1:04:09
Yeah.
Um.

Tarun Jain 1:04:13
So where is the context coming from? Context is coming from search. So what is search?

Hardip Patel 1:04:18
Right.
OK, oh OK, I I guess docs dot context.

Tarun Jain 1:04:27
Correct. So if you notice what is the purpose of join? If you have anything in list, convert that into string. Is this clear? But now what is the data type of docs?

Hardip Patel 1:04:35
Mhm.
Mhm.

Tarun Jain 1:04:43
It is in dictionary, correct? But where is the list? The list is inside context, so I just have to do docs of context.

Hardip Patel 1:04:46
Yeah.

Tarun Jain 1:04:53
So now this is in string. You have the prompt prompt as context and user is nothing but the query.

Hardip Patel 1:05:02
Hmm.

Tarun Jain 1:05:03

Now what do we define here? I will define my client equals to LLM client.
And this will be LLM client dot get response.
And I don't need to have to do content, I will just return answer.
So now if I do testing.
Mail to contact.
That's it. This is your rag. So you have search which will get the documents. Then you have answer where you had your user query. Inside user query you're getting the relevant documents, then asking a prompt and then generating the response. So now if I print this.
I should have the actual LLM response.

Hardip Patel 1:05:52

Uh.

Tarun Jain 1:06:02

You can mail Atyantik at contact at Atyantik.com.

Hardip Patel 1:06:07

Mhm.

Tarun Jain 1:06:10

Is this clear? We didn't try any new code, we are just testing it in multiple files. That's it.

Hardip Patel 1:06:16

Hmm.

Tarun Jain 1:06:18

So we have to make one change here. Something felt wrong. Huh. So you'll also have one more files. One more file is constant dot PY. So constant constant dot PY if there is any variables like prompt.

Hardip Patel 1:06:25
Hmm.

Tarun Jain 1:06:34
So instead of this, let me rename. I'll make it as prompts dot PY. This is only when if you're working with LLM based application. Here you will have your system prompt. And you love your user.

Hardip Patel 1:06:48
Uh.
Uh.

Ajay Patel 1:06:57
Uh, we can also be edit in the config now if we want to.

Tarun Jain 1:07:01
Even that is possible here also you can add, but if you had that here it will be like too lengthy.

Ajay Patel 1:07:02
One.

Hardip Patel 1:07:06
Um.

Ajay Patel 1:07:07
Yeah, yeah, makes sense.

Tarun Jain 1:07:09
Because here if you add prompts dot PY you know exactly where to have to change and this will be like 1010 lines, 2020 lines in some cases.
If you have system prompt user prompt which is just one line, then you can add in config dot PY which is much easier.

Ajay Patel 1:07:19
Um.

Tarun Jain 1:07:26
But if you have more than 2-3 lines, then you can keep it separate.
So we completed client dot PY, config dot PY and state dot PY. So let me just check if you got the output or not. You guys got the output.

Hardip Patel 1:07:46
Uh, I I have to work on answer.

Tarun Jain 1:07:52
Oh, what happened?

Hardip Patel 1:07:52
OK, sorry you you pasted the I was working on the answer. No problem. Sorry.

Tarun Jain 1:07:58
OK, the tensor is there in this collab. It's only like collab in bigger side.

Hardip Patel 1:08:01
Yeah, I got it, got it.

Tarun Jain 1:08:05
So now what changes are we supposed to make here in state dot PY?
This will be back to.
State which is rack state.

Hardip Patel 1:08:16
Hmm.

Tarun Jain 1:08:23
And whatever your query is there, where did this query will come from state of query?

Hardip Patel 1:08:30
Mhm.

Tarun Jain 1:08:30
This should be state of query.
Then anything else is there. No, that's it. And what are we returning? We are returning, returning in a dictionary as we saw yesterday. So instead of state, let's follow the document format where we are returning dictionary. This will also become state.
Rag state.
And will we pass docs here? Will we pass this?

Hardip Patel 1:08:59
No, it will be not.

Tarun Jain 1:09:01
No. So what should we do? We already have state, so this will be state of context.

Hardip Patel 1:09:08
Oh.

Tarun Jain 1:09:14
This will be context. This will be state of query and I'm returning answer inside my main.
Or what I'll do is create a function called.
Workflow.
And what are we supposed to do now? Copy this? Where is class rack state?
Last rack state copy this year at the top.
So we don't need answer relevance, we don't need context relevance. So where do we get these two things from?

Hardip Patel 1:09:51
Yeah.
Identic.

Tarun Jain 1:10:00
Hey, these two things.


Hardip Patel 1:10:03
Uh oh, Nampai.


Ajay Patel 1:10:07
Not numpy. Not numpy is not there.

Tarun Jain 1:10:08
Oh.


Hardip Patel 1:10:08
I.


Ajay Patel 1:10:13
Data not a data set.


Hardip Patel 1:10:16
But.


Ajay Patel 1:10:19
Data set.
That was another similar.

Tarun Jain 1:10:25
So what is this this called this syntax?
What is this called?


Ayush Makwana 1:10:36
It's typing I think.


Mitesh Rathod 1:10:37

Function return typing, yeah.
And.

Tarun Jain 1:10:40
One said the keyword.

Mitesh Rathod 1:10:41
Function typing.

Ayush Makwana 1:10:42
You are typing.

Tarun Jain 1:10:43
Typing.

Hardip Patel 1:10:44
It bings OK, OK.

Tarun Jain 1:10:46
Import.

Mitesh Rathod 1:10:49
Type it.

Tarun Jain 1:10:49
Type comma list.

Mitesh Rathod 1:10:51
It's OK.

Tarun Jain 1:10:54
So we are not writing new code, we are just copy pasting from here.
So this line.
Now what is the another line we need? We need land graph start and end.
So let's just divide. Uh, this is.

Which is the files from your own code. You have to keep that at the second. So typing is default which is predefined. So I will copy this. I will paste it at the first line, then client it is from your own code, config it is from own code. Then whatever package you have, you can keep that in one group.

Hardip Patel 1:11:29

Mhm.

Mhm.

Tarun Jain 1:11:32

So now this is like predefined.

This is your own code which is from code files and then this is like uh external libraries where you have to install and get it.

Hardip Patel 1:11:39

Mm.

Tarun Jain 1:11:49

So now what we are supposed to do? We need to define the state graph.

Why do? I'll just copy.

Where did we define workflow? Ha, this one. So now inside workflow.

So instead of workflow this thing, what we can do is we can do graph builder.

Graph builder then workflow.

Workflow dot add node. So is this search context or is it search?

Did you search?

Hardip Patel 1:12:34

Oh.

Tarun Jain 1:12:36

Add search.

Hardip Patel 1:12:37

No.

Tarun Jain 1:12:38
Then this is answer. I'll make it answer.

Hardip Patel 1:12:44
Um.

Tarun Jain 1:12:47
Then this is search. This is answer. This is answer.

Hardip Patel 1:12:51
Oh.

Tarun Jain 1:12:59
Now what will I return? I will just return graph.

Hardip Patel 1:13:02
Oh.

Tarun Jain 1:13:08
This is the new port that we have written this this particular cell, this class and this import statement.
Now where do you think I need to run this graph builder? In which file?

Hardip Patel 1:13:28
Main.

Tarun Jain 1:13:29
Correct. So in main dot PY, what will I do now from state?

Hardip Patel 1:13:30
Oh.

Tarun Jain 1:13:36
I will import.

Graph builder. Now here I will have.
User input. So whatever UI you have to create, you will create it here.
Streamed it UI.

Hardip Patel 1:13:50
Mhm.

TJ

Tarun Jain 1:13:52
So what I'll do is I'll ask user input enter.
You're wrong.
So let's run a for loop while true.
So until user enters quit, I want to keep asking questions. So how will my logic look like?
You understood. I want user to ask their input as much as possible, but as soon as it sees queue or quit, you should end it. So what will be the logic?

Hardip Patel 1:14:19
OK.

TJ

Tarun Jain 1:14:29
Inside while true, what will be the first line?

Ajay Patel 1:14:34
Initiate the first.

Mitesh Rathod 1:14:37
Input.

TJ

Tarun Jain 1:14:37
You love user.

Ishan Chavda 1:14:39
You can listen in.

Tarun Jain 1:14:39
User prompt equal to input enter your query.

Hardip Patel 1:14:43
Yeah.

Tarun Jain 1:14:46
So if I tell width or Q to exit, now what this line will be?
This is very tricky. If someone tells this, I'll give you one swag when I come. Definitely I'll give. We should not use any autocomplete option. We should not use autocomplete.

Hardip Patel 1:15:00
It.

Ajay Patel 1:15:04
Sorry, son.
1.

Hardip Patel 1:15:10
Starts starts with user run dot starts with.

Mitesh Rathod 1:15:15
No, no, no, no, no, no. Lower case, lower case.

Tarun Jain 1:15:17
I will not write it here. I will write it here because there you have auto complete.

Mitesh Rathod 1:15:22
No we we can use a like a input then convert into the lower case and then right exit.

Tarun Jain 1:15:27
Oh.

Mitesh Rathod 1:15:30
Mitesh.

Tarun Jain 1:15:31
Correct. So I've.
You will have if.
User prompt dot lower.

Mitesh Rathod 1:15:39
Lower, yeah, FTRS that fit, yeah.

Tarun Jain 1:15:42
Then starts with.

Ajay Patel 1:15:44
That sweet.

Tarun Jain 1:15:46
Now if it is Q, you will just stop it.
Or it's better we it.

Mitesh Rathod 1:15:53
But it's it can be anything right? Yeah, for the queue. But if we can just exactly match like a double equal to or triple equal to.

Ajay Patel 1:15:55
Yeah.

Hardip Patel 1:15:58
Thank you, Sir.

Mitesh Rathod 1:16:04
That will be, I guess, better.

Ajay Patel 1:16:04
Nobody.

Tarun Jain 1:16:04
Hi.
Double equals to quit.
Or userprompt dot lower double equals to Q.

Hardip Patel 1:16:22
M.

Tarun Jain 1:16:26
You will break. If not, you will just add graph builder and user prompt. That's it. So now I'll just run this.
Python main.py.
Enter your query. I'll just add QUI.
Takes one positional argument. Oh, it should be dot invoke.

Ajay Patel 1:16:50
That one was given.
Invoke.

Tarun Jain 1:16:55
This is response equals to graph builder. So what is graph builder returning? It is returning graph. So how do I run this? It should be like query of user prompt.

Ajay Patel 1:17:02
Hmm.

Tarun Jain 1:17:11
Is this clear? And then print results.

Ajay Patel 1:17:12
Hmm.

Tarun Jain 1:17:15
I know if I come back.
Python main dot PY. If I do quit, it stops. Let me have one function. Print. Thank you for using our service.
But thank you for using our service then Python dot PY. How do I contact?

Hardip Patel 1:17:48
Graph.

Tarun Jain 1:17:48
OK, why is it?

Ajay Patel 1:17:51
Function object.

Tarun Jain 1:17:53
I did it. Oh no.

Ajay Patel 1:17:56
Work.

Hardip Patel 1:17:59
I will not come by graph. I think it's fine. Um.

Ajay Patel 1:18:05
Yeah.

Tarun Jain 1:18:07
Graph dot invoke this graph is nothing but compile.

Hardip Patel 1:18:12
Um.
Oh.

Invoke.
OK.

Tarun Jain 1:18:30
Why is it?

Hardip Patel 1:18:31
Sorry.
Can we can we add a return type?

Tarun Jain 1:18:37
No, it's the return graph.

Hardip Patel 1:18:39
Yeah, no, the the return type of OK.

Tarun Jain 1:18:45
State graph of defined drag state is defined.

Hardip Patel 1:18:48
Mhm.

Tarun Jain 1:18:50
Add not search function is there.
Search it has state rag state state rag state F.

Hardip Patel 1:19:01
What is the error?

Tarun Jain 1:19:08
The function object has no invoke, so we test here if name.

Mitesh Rathod 1:19:13
Can we write down to like a type of type of the graph variable?

Tarun Jain 1:19:21
Oh ****.
This is a function.

Hardip Patel 1:19:25
Oh, OK.

Mitesh Rathod 1:19:25
Function correct.

Tarun Jain 1:19:32
Python main dot PY.
How do I contact?
Why is it this lengthy response?

Ajay Patel 1:19:54
M.

Tarun Jain 1:19:54
OK, so here it should be answered.

Hardip Patel 1:19:58
Hmm.

Tarun Jain 1:20:01
And we are getting one warning message. If you see this warning message is from.
So if you want to remove the warning message, import warnings.

Hardip Patel 1:20:04
Yes.

Tarun Jain 1:20:15
Warnings dot.

Filter warnings ignore.
Yeah, now this is your main dot PY.

Hardip Patel 1:20:30
Yeah.

Tarun Jain 1:20:42
So whatever stream code you have, you will add it here.
So client dot PY is done, config is done, main is done.
Models we don't have. So for timing what I'll do is I will delete models but I I hope you understood when to use models. Whenever you are defining output parsers you will have schema. During that time we'll define models dot PY. I will delete this since we don't have it.
Then prompts dot PY is there. Then you have state dot PY, then utils dot PY. If in case we need to do any error handling, we will add that here.
All right, eval is pending. We'll come to eval for time being. Can you test this if it is working?

Hardip Patel 1:21:26
Um.

Tarun Jain 1:21:30
What clear?

Hardip Patel 1:21:32
Yeah.

Tarun Jain 1:21:34
Python window device. What is not you?

Hardip Patel 1:21:36
OK, can you send a state, a state class?

Tarun Jain 1:21:42
One second.

Hardip Patel 1:21:43
Hmm.

Tarun Jain 1:21:46
So now if you see there is no warning message.
Oh, still it's there.
Authentic users help with framework. The context does not provide specific name of this framework.
What is the text tag?
Use.
Are these.
GCP interested.
I don't think you use Python, right?

Hardip Patel 1:22:36
I don't know.

1:22:40
Maybe I I guess.

Hardip Patel 1:22:42
We use in back office, so maybe.

Tarun Jain 1:22:47
OK, what is it mentioned in the website URLs?

Ajay Patel 1:22:48
But.
Yeah, it it is mentioned. It is mentioned.

Tarun Jain 1:22:53
Cool, then fine. So what?

Hardip Patel 1:22:54

Tarun can the state code again because we added the graph builder and I wasn't able to follow up.

Tarun Jain 1:23:03
So graph builder, once you define in main function we have to use this function. This is what we missed. So this is the new code.

Hardip Patel 1:23:10
Yeah.

Tarun Jain 1:23:17
So this should come in state dot PY.

Hardip Patel 1:23:22
Even answer I think has been updated.

Tarun Jain 1:23:28
No answer I didn't update. Answer is same.

Hardip Patel 1:23:32
Uh.

Tarun Jain 1:23:33
So the only thing we changed was we added this thing. This was missing.

Hardip Patel 1:23:40
Very good. It was same, right? Yeah.

Tarun Jain 1:23:40
I'll share it.

Hardip Patel 1:23:45
Yeah, I think that will just happen.

Tarun Jain 1:23:46
So how many of you got the results?


Ayush Makwana 1:23:53
Yeah, I got the result.

Tarun Jain 1:23:58
Well, Evals will continue tomorrow. It's the same thing. Evals. Now what we will do is if not, it will take 5 minutes.
So copy this code, import pandas, paste it here.
Open this, paste it here. Here create a new folder called data. Inside this data, let's add that CSV file.
So this will be data Ragha's data set, so let me copy that.
So I have data folder.


Hardip Patel 1:24:36
OK.

Tarun Jain 1:24:39
Desktop evals.
Inside data I need to have raga's data set.
Where is it Raga's data set?
And what is the next line of code I have to run this?
Then.


Hardip Patel 1:25:13
Thanks.

Tarun Jain 1:25:19
It made out for PY. Everyone have completed. You got the results.
OK, so now I'm just using eval dot PY same code what we wrote in the morning.


Hardip Patel 1:25:24
Yeah.

Ayush Makwana 1:25:24
Yeah.

Tarun Jain 1:25:31
Copy this, paste it here, then copy this.
Paste it here. So this data is nothing but DF and then I will print.
So I'm just testing if it is working in collab or not.
Python eval dot PY.
Can you say I got the result?

Hardip Patel 1:26:34
Oh.

Tarun Jain 1:26:36
Events data set. Then you have features, user input, reference, context and reference.
What are we supposed to do here now?
Before I pass it to eval data set, what am I supposed to do here at this particular?
Here we need to apply some logic. What are we supposed to do?

Mitesh Rathod 1:27:00
Uh, regarding.

Ajay Patel 1:27:01
What he was.

Tarun Jain 1:27:03
So if you see here how many input variables we have.
You have three. How much is it supposed to be?

Ajay Patel 1:27:10
Big 3.

Mitesh Rathod 1:27:11
Um.

Tarun Jain 1:27:14
So total variables we need total 5. So the two pending are retrieve context and.

Mitesh Rathod 1:27:15
The number of the columns.

Tarun Jain 1:27:23
What is it called? Retrieve context and response?

Hardip Patel 1:27:23
But.

Mitesh Rathod 1:27:24
Fantasize her name. Fantasize her name.

Hardip Patel 1:27:25
OK.

Tarun Jain 1:27:28
So what I'll do is if you see here I have a function called get rag response. I will copy this. I will paste it here. This is from yesterday's code.

Hardip Patel 1:27:36
No.

Tarun Jain 1:27:40
Now here what we need to do is yesterday we had data eval sample inside that you had response then you are taking response dot context. So now can anyone tell me what will change here?
I will not have any inputs earlier.
So for data in DF.
Then you'll have DF dot.
User input.

Hardip Patel 1:28:15
Mm.

Tarun Jain 1:28:15
This is your query right? If you see data of user input is where you have your data so you can check this. I will copy this line.

Hardip Patel 1:28:16
Yeah.

Tarun Jain 1:28:25
I will paste it.
Here so for data in DF.
OK, this is data.
For info keep it.
And now if I print query.

Hardip Patel 1:28:46
OK.
I will have to OK.
The the instance restarted so.

Tarun Jain 1:28:59
No, it's still running.

Hardip Patel 1:29:02
It was connecting, so maybe.

Tarun Jain 1:29:08
Data for friend data.
For print info.

Hardip Patel 1:29:15
OK.

Tarun Jain 1:29:20
User input.
So for data in user input.
I have all this for you. OK, this is what I need to run.

Hardip Patel 1:29:31
Oh.

Tarun Jain 1:29:37
So this is DF. Now this info is where you have your query.

Hardip Patel 1:29:42
Mhm.

Tarun Jain 1:29:43
So now what will happen here?

Hardip Patel 1:29:47
Here uh, we'll need to import the state dot.

Tarun Jain 1:29:54
Correct. We have to import from state from state import graph builder. I'll quickly complete this. We are on time graph builder dot invoke.

Mitesh Rathod 1:29:56
Graph leader.

Hardip Patel 1:29:57
Yeah.
OK.

Mitesh Rathod 1:30:02
That's very good.

Tarun Jain 1:30:14
Invoke.

Mitesh Rathod 1:30:14
OK.

Tarun Jain 1:30:16
So what I'll do is I'll define graph equals to graph filter.
Now this is much simpler graph dot invoke.
Query.

Hardip Patel 1:30:29
Hmm.

Tarun Jain 1:30:29
And it should be query query and then what you're supposed to do is get the retrieve context.

Hardip Patel 1:30:39
Mhm.
So.

Tarun Jain 1:30:52
Table you'll get an error and we have seen this in pandas. So whenever you're appending in the pandas you should have the same length. So now what will I do here? I will just do response dot append.

Hardip Patel 1:30:52
So.

Tarun Jain 1:31:08
Response dot answer.

Hardip Patel 1:31:10
Oh.

Tarun Jain 1:31:12
And this will be retrieved context.
dot append response dot context and then what will I do?

Hardip Patel 1:31:20
It.
Hmm.
Uh, I.

Tarun Jain 1:31:28
V dot retrieve context. VF dot response is equals to response. Then you have eval data set. Now if I print eval data set.

Hardip Patel 1:31:31
Oh.

Tarun Jain 1:31:39
I should have total uh 5 what you call 5 features.
Python eval dot PY. This will take time.

Hardip Patel 1:31:47
Right.

Tarun Jain 1:31:54
Where is it throwing error?
Oh, I didn't call this function.

Hardip Patel 1:32:02
1.

Tarun Jain 1:32:03
So let me remove this run directly.

Hardip Patel 1:32:14
Yes.
OK.

Tarun Jain 1:32:16
Yeah, until then anyone has any doubt or if anyone is not getting output in main dot PY, you can let me know. Let's just refactor this, keep it as graph equals to graph builder and then graph dot invoke.

Hardip Patel 1:32:24
Hello.
Mm.

Tarun Jain 1:32:32
Remaining everything is same. This will take time if it is 2 seconds for every question, so this should take at least 30 seconds.
Response hot.
Oh, this response and this response is same, so I'll keep this as a result.

Hardip Patel 1:32:54
Oh.

Tarun Jain 1:32:55
Result answer result graphbuilder graph invoke query query.
Then response retrieve one text.

Hardip Patel 1:33:09
Yeah.
Oh, OK.

Tarun Jain 1:33:18
Uh, did anyone try?

Hardip Patel 1:33:19
She.
Uh, no, I'm not following currently.

Tarun Jain 1:33:24
OK, but till main dot PY everyone have done right?

Hardip Patel 1:33:27
Yes.

1:33:28
Yes.

Tarun Jain 1:33:28
OK, So what we can do is we can just create a folder data inside that add ragas data set. I will send you the code.

Hardip Patel 1:33:34
Mhm.

Tarun Jain 1:33:37
And there is no new here, so I hope you understood the syntax still here. Here what we are trying to do is for every user query we need to get context, we need to get response of equal length. That's the reason why I have list and once I have the only purpose is to have.

Hardip Patel 1:33:42
Yes.
Oh.

Tarun Jain 1:33:53
5 features and then what am I supposed to? I will just run this function.

Hardip Patel 1:33:59
Oh.

Tarun Jain 1:34:03
Print.
Result.
And this evaluate is coming from these two lines.

Hardip Patel 1:34:13
Oh.

Tarun Jain 1:34:16
And for evaluator LLM I need to add open a API key.

Hardip Patel 1:34:20
Oh.

Tarun Jain 1:34:28
So I'll just do from line.

Hardip Patel 1:34:30
On a figure.

Tarun Jain 1:34:34
From config.
Import configure.

Hardip Patel 1:34:48
Oh.
Yeah.
Nice. Hmm.

Tarun Jain 1:35:05
Everyone understood I'm using open AIP key then result and then evaluate. So I hope I have the five response. So can you see so evaluation data set I have user input, retrieve context, reference context, response and reference.

Hardip Patel 1:35:08
Um.

Tarun Jain 1:35:22
So till here it is working. Now I'll just run this code and before that I have to paste open AI API key.

Hardip Patel 1:35:25
Mhm.

Tarun Jain 1:35:41
And here, that's it. So now if I print this, I should get the final result. That's it.
So we didn't write any new code, the only thing is we have it in separate file.
You understood this is the new code we added which is evaluate and apart from this evaluation data set need 5 variables when you run evaluate which is user input, reference context, reference, retrieve context and response. So this two is from LLM.
Remaining other thing is from the domain expert.
When you generate the data set.

Hardip Patel 1:36:24
Hello.

Tarun Jain 1:36:29
So it should take one more minute and we'll have output.
So evaluation data set is done.

Hardip Patel 1:36:41
Mm.

Tarun Jain 1:36:42
Now it is running ragas.
So this is 18 zero to 18.

Hardip Patel 1:37:07
Yeah.
So for VS code, is there a better way to so normally like when when I'm doing the source VNV the interpreter is not taking in so there are lots of warnings.

Tarun Jain 1:37:30
Yeah, this warnings we have to avoid. I'm not sure why we are this GC with them.

Hardip Patel 1:37:33
No, no, not.

Tarun Jain 1:37:38
Then.

Hardip Patel 1:37:39
So in the code like a then it does not pick up the interpreter the Python and so like whatever PP install I will do it will not take in the ID I mean and it the nothing will work so by for.
PHP storm I can select the interpreter, but for VS code I don't know how to do that.
I don't know if I make sense in the for meaning code.

Tarun Jain 1:38:07
Probably we can have a phone call till here. Everyone had done it. I got the output.
Faithfulness is 100%. Factual correctness is 72%. Context recall is 79%. This is.

Hardip Patel 1:38:14
Yeah, no. OK, OK.

Ayush Makwana 1:38:17
Yes.

Tarun Jain 1:38:22
Perfect. So if you have more than 70%, that is good enough. If you have more than 80%, that's even good. More than 90% your app is you have to cross it because more than 90 it's impossible.
Even have to check here. I don't know how I got 100%. This is nearly impossible.

Hardip Patel 1:38:38
OK.

Tarun Jain 1:38:44
So I have to cross-check this, but these two scores are correct.

Hardip Patel 1:38:51
Mm.

Tarun Jain 1:38:53
So I'll just copy this and I will paste it here. For this you need open API key but what you guys can do is here add from ragas.
dot LLMS import.
Line chain wrapper.
Since we are not using what you call data set creation, here you can use LLM equals to line chain wrapper and inside this you will have.
Where is the plan dot PY?
Client dot PY.
Copy this.
You are space to tear.
From lunching.
Google.
So this is for them who don't have open EAFPA key. You can try this and you can let me know if you got the output or not.
Yeah, Adi will connect regarding the VS4 thing.

Hardip Patel 1:40:07
OK, OK.

Tarun Jain 1:40:08
Anyone has any doubt? I hope you followed how to use the.


Ajay Patel 1:40:11
Yeah, yeah.

Tarun Jain 1:40:13
The only thing is this is new. Remaining is we checked it already yesterday.


Ayush Makwana 1:40:15
Yeah.


Ajay Patel 1:40:15
M.
OK.


Hardip Patel 1:40:21
Hmm.

Tarun Jain 1:40:21
So try again. Evals is something that is sometime it can become tricky. So what you can do is first write on collab because you you saw it for this evaluation as I run two times and here I have only 10 questions. Sorry 6 question.


Hardip Patel 1:40:32
Mhm.
Oh.
Hmm.
Hmm.

Tarun Jain 1:40:55
Sorry, collab.
Run evals on collab so that you don't have to repeat this multiple times. This is the only issue this particular code cell because once you have this thing right either what

you can do is save the CS sphere.
Save the CSV.

Hardip Patel 1:41:15
Mhm.

Tarun Jain 1:41:15
After this separate just do DF dot to CSV and this will be our final CSV.
dot CSV file then index equals to false. Once you have this file then come to VS code.
Before this run everything on collab.
Once you have this final CSV file then what you can do is then you can have VS code for the evaluate function.

Hardip Patel 1:41:44
OK.

Tarun Jain 1:41:45
So now how this will look like? Let's suppose this is final response and assuming you have run till here on Colab. Now what you will do is here you will have import pandas as PD.
Then uh PD dot CSV read response, which is your final thing.

Hardip Patel 1:42:06
Mm.

Tarun Jain 1:42:07
Then what you're supposed to do is DF dot reference context. You have to change this into list. Even retrieve context will be in dictionary. I mean string you have to convert that into.

Hardip Patel 1:42:11
OK.
OK.
List.

Tarun Jain 1:42:22

List.

Apply eval.

And then when you do eval data set, evaluation data set, whatever DF is there, paste the DF here then remaining code is same.

This will be on VS code before this, which is our final response CSV. This will be on collab. Is this clear?

Ajay Patel 1:42:52

M.

Hardip Patel 1:42:53

Mm.

Tarun Jain 1:42:56

Below code.

On VS code.

This is on collab.

Good.

Yeah, that's it for today.

I hope you understood the logic of whatever we did.

Hardip Patel 1:43:18

Yes.

Ajay Patel 1:43:18

M.

Tarun Jain 1:43:20

Well.

Any questions?

Ayush Makwana 1:43:23

OK.

Ajay Patel 1:43:23

No so far.

Ayush Makwana 1:43:25

Um.

Hardip Patel 1:43:26

No, thanks for taking time to look at the code by the way.

Tarun Jain 1:43:30

Yeah, if anyone has completed, you can mail me the code. I will check it out.

Yeah. Thanks everyone.

Ajay Patel 1:43:38

Thank you. Thank you.

Ayush Makwana 1:43:39

Thank you.

Ishan Chavda 1:43:40

Thank you.

Ishan Chavda stopped transcription