

Python and AI Power-Up Program Offline Class- 20250925_113742-Meeting Recording

September 25, 2025, 6:07AM

1h 37m 7s

- **Tirth** started transcription

 **TJ** Tarun Jain 0:04

So did you guys try this? Like did you get the result from eval dot PY?

 **Ronak Makwana** 0:11

Sorry, but I have some other priorities work so I didn't try.

 **TJ** Tarun Jain 0:18

OK, so anyone got more than 80% in this particular evals dot PY?

 **Ayush Makwana** 0:21

Uh huh. Yeah.

Oh, actually with this Gemini, I hit the rate limit multiple times, so I didn't try it then.

 **TJ** Tarun Jain 0:33

OK, so here you added LLM evaluator, right? LLM parameter and evaluator LLM was Gemini.

 **Ayush Makwana** 0:38

Mm.

Yeah, yeah.

 **TJ** Tarun Jain 0:42

OK.

Uh, did anyone try with Open UI?

 **Hardip Patel** 0:46

I I won't be able to try till weekend, but on weekend I will do, yeah.

 **Tarun Jain** 0:53

OK, but you were able to run the main dot PY right?

 **Hardip Patel** 0:56

Yeah, I was.

 **Ayush Makwana** 0:56

Yeah, yeah.

 **Tarun Jain** 0:57

Everyone got the result in main dot PUI.

 **Hardip Patel** 1:02

But just to mention if anyone wants the Azure, I can share the API key just need to use wisely.

 **Tarun Jain** 1:10

So for Azure, there are three things you need to provide. Uh, one will be.

Azure endpoint and then you'll have Azure API key.

And then we'll have deployment uh name.

So these are the three things we'll be needing. So OS dot environment you need to add Azure endpoint, then Azure API key deployment name is nothing but your model name.

So this has to be deployed.

So everyone has Azure or just are the only you have it?

 **Hardip Patel** 1:53

Yeah, he's already with me, so that won't be a problem.

 **Tarun Jain** 2:01

So now I mean as your deployment, is it there with everyone or just with one or two folks?



Hardip Patel 2:02

Sorry.



Ronak Makwana 2:07

Yeah, I also registered yesterday because I am also hitting the limit.



Tirth 2:12

I also have uh Azure so I can share it with everyone. I have \$800 pending as credit.



Tarun Jain 2:20

OK, and probably let me show you something on how to create the deployment. So what you guys can do is we can note down the steps.



Tirth 2:35

OK, I think we are recording, so that should be fine as well, yeah.



Tarun Jain 2:38

OK, so the first step what you need to do is you just have to search for Azure Open AI and once you click on Azure Open AI, this is how your screen will look like. You'll have AI Foundry and inside AI Foundry you will have Azure Open AI. So now what you need to do is if you're starting.



Tirth 2:44

Yeah.

Mm.



Tarun Jain 2:57

You have to click on create and once you click on create you have to create a new resource group. If you have a resource group available you can pick the existing ones. So for example I'll pick the existing one. Then what you need to do is you can give any name so.

I'll give testing and once you click on testing you can select standard 0.



Tirth 3:21

Just just to tell everyone like use the region E US only because I think many models and everything they are supported in East US only.

 **TJ** Tarun Jain 3:22

What is?

Huh. The pricing was is also less in East US.

 **Tirth** 3:34

Yes.

 **TJ** Tarun Jain 3:35

So name I'll give I'll not create the resource group, I'm just giving it a name. So this is the first step you need to create the resource group. Once you create the resource group you can just click on next next and then it will be created.

 **Tirth** 3:36

OK.

 **TJ** Tarun Jain 3:54

So I'll come back. As soon as you create your resource group, you will have a new entry. So this new entry will be at the antic. Now if I click on this.

So I'll bring this here.

So first is Azure Open AI from Azure Foundry.

And then resource group.

So as soon as the resource group is created here you will have something called as endpoints. So if you click on this endpoints you will have your key and you will have your endpoint. So these two are very important. One is key and one more is endpoint. So now you have key.

And end point as soon as you create a resource group and the last part what you need is you need the deployment name. So deployment name we have models like GPT photo. Then if you want to use reasoning models you can use O4 mini.

If not, you can also use O3O3 will be costly, so I will prefer. If you want reasoning models, you can go with O4 mini. GPT 4O is good and GPT 4.



Tirth 5:11

GPT5GPT5 does not have research partner.



Tarun Jain 5:16

Uh, which one?



Tirth 5:17

GBT 5 in open in Azure.



Tarun Jain 5:19

GB5I didn't switch, I just tested it. Since it was a bit costly, we didn't use it anywhere.



Tirth 5:22

OK.

OK, welcome.



Tarun Jain 5:26

But I can't comment. The results are fine, but we are still getting better results with GPT 4.1.



Tirth 5:35

Perfect. OK.



Tarun Jain 5:36

And yesterday I also got to know that temperature is not supported in GPT 5 temperature top P, top K and all.



Tirth 5:43

Right.



Tarun Jain 5:46

So now how do we create this deployment group? You can come back to your resource group and as soon as you click on resource group.

I don't know why this is slow. Huh. You'll see a button here called Explore Azure AI

Foundry Portal. Now I'll just click on this explore.

So this will open a dashboard where you'll have Azure AI Foundry.

Now here what you can do is as soon as you open on the left hand side you have deployments. So whatever deployments you have created right it will be here. So as you can see you have GPT 4O and you have O4 mini the same models which I mentioned here. So now what you can do is you can click on deploy model.

Deploy base model.

So you can select which model you need. For example, let's suppose I pick 04 mini.

So now this 04 mini is my deployment name, so when I click on confirm.

So since I already have 04 mini, what it has done is the deployment it created is as 04 mini 2. So here you can also add like at the antic 04 mini. So this can be any name, but usually it's always best practice to keep the same model name.

So since it's already created, I'll keep it too. Then this can be global standard. Once this is done, you can click on deploy and if in case you want to change this, let's suppose how many tokens per minute you need. If you are running evals and if it is utilizing more tokens, what you can do is you can increase this to.

1,000,000 or anything. So now if I click on customize I will increase this to 1,000,000. And then I'll click on deploy. That's it.

So now you have your API key, you have your endpoint and then you have your deployment name. So once you have these three details how we used Lan Chain, Google Gen. AI, you just have to search for Lan Chain.

Open AI.

And if you click this here you'll see Azure Open AI.

How's your open UI land chain?

So this is the one you have Azure Open API key, then Azure Open AI endpoint. So this endpoint you have to copy then also copy your API key and now if you scroll below you just have to import from Lanchen Open AI, import Azure Chat Open AI. So deployment name is nothing but what we had earlier. So when you create a deployment name you will have the deployment name. API version is nothing but this thing model version. Just copy this, paste it here and then remaining things are same. So the only thing is instead of model name you will add it as Azure deployment.

And copy whatever you see here.

Oh, is this clear? So this is for Azure Open AI.



Tirth 9:05

Yes, very clear.



Ronak Makwana 9:05

Yes.



Tarun Jain 9:13

Yeah, cool.

So I have the code here only, so whatever we did yesterday. So you have client dot PY here you can just replace Google generative AI to Azure Open AI.

And then evals is something that I don't think many of you ran. So this is once you have the results, you can let me know and I'll give one assignment at the end.

Probably we'll start with today's session. And this was also one interesting assignment which Ayush submitted yesterday. So he created this transcript chatbot. So whatever session.



Tirth 10:04

Yes, you know.



Ayush Makwana 10:05

You know.



Tarun Jain 10:05

She's there, yeah, in call.



Ayush Makwana 10:07

Yeah.



Tarun Jain 10:08

So instead of upsert, you have something called as upload back search.



Ayush Makwana 10:17

OK.

 **Tarun Jain** 10:17

OK, I didn't run it. So if you just search for quadrant client.

Upload points.

So usually we do upset right instead of upset if you have batches.

So this is what we used. We used client dot Upset. There is also one more command for batch.

 **Ayush Makwana** 10:37

Right.

 **Tarun Jain** 10:43

Load patch.

What?

OK, where is that function? I'll send you that function. You have something on upload. So instead of upset, what you can do is you can use upload if you have batches.

 **Ayush Makwana** 11:18

OK.

 **Tarun Jain** 11:19

Remaining things are same. Whatever was there, it was correct only. Only this one change. Probably since you have too much of data right, it takes too much of time.

So when I saw this, probably I understand did it crash there on call?

 **Ayush Makwana** 11:21

OK.

Yeah, so yeah, yeah, without this kind of batch processing, it was giving me this HTTP timeout error when I tried to update it.

 **Tarun Jain** 11:42

And here also you missed out the binary quantization.

 **Ayush Makwana** 11:48

Uh, cool.

 **Tarun Jain** 11:50

So one is binary quantization you can add.

 **Ayush Makwana** 11:54

OK.

 **Tarun Jain** 11:55

When you create index. So now what you can do is since you already created index, you can take three more days of transcript and then you can index it again. So one is binary quantization and then you can use upload instead of upsert.

 **Ayush Makwana** 12:03

Hmm.

 **Tarun Jain** 12:13

If.

If you want to use batching.

And yeah, this script was correct only app dot PY.

 **Ayush Makwana** 12:26

Yeah.

 **Tarun Jain** 12:30

Pulia.

 **Ayush Makwana** 12:31

I want to just ask that I send you the PDF as well, the PDF of the transcript. Yeah, so is there any way we can preprocess it? Because when you open it.

 **Tarun Jain** 12:37

Yeah, it's OK.

This one it.



Ayush Makwana 12:51

Yeah, so I have data like this like Tarun Jain 042 and then your dialogue then all. So is there any way we can do more structure processing and give it to the other network embeddings?



Tarun Jain 13:09

So what do you have in metadata?



Ayush Makwana 13:13

In metadata I am like I don't have pages or page number so I create one chunk and assume it as a like page or something.



Tarun Jain 13:26

So what I was thinking when I saw his data is like this details are there right name and timestamp. This can be the metadata and then whatever you have here is this is your page content.



Ayush Makwana 13:32

Yeah.

OK.



Tarun Jain 13:43

So here what you need to do is since you have your own custom this thing right, you have your own custom page content also. So usually what happens you'll have your own custom metadata, but in your case this is your page content and this itself is custom.



Ayush Makwana 13:51

Hmm.



Tarun Jain 13:58

So what you can do is directly you can use document schema of langchain. So what

is this document schema? I'll just explain. This is also important for others if you want to use your own custom content and then convert that into langchain format.

 **Ayush Makwana** 13:59

Hmm.

 **Tarun Jain** 14:18

So what I'm trying to say is you have raw data after pre-processing, but what does line chain need?

So if you want to use chunking and if you want to use embedding, this has to be in document format. So this document format needs to be like page content and metadata.

So if you want to convert your raw text to document, what you have to do is there is line change schema line change document.

So you just have to pass like this.

Right here it was good. So now if you see what is your page content, page content is nothing but whatever raw data you have here and metadata is nothing but whatever you have extracted which is name and timestamp.

 **Ayush Makwana** 15:14

Hmm.

 **Tarun Jain** 15:20

So here you will have name. You can also have source. So author instead of author you will have person name which will be Tarun, Teer, Adip, anything. And apart from that you can add one more thing which is timestamp.

 **Ayush Makwana** 15:20

Hmm.

 **Tarun Jain** 15:36

Oh, you got it here.

 **Ayush Makwana** 15:37

Yeah OK yeah I got it, got it. And also one other thing, when I try to deploy this on

Streamlit, it is giving me this health check error or something with like let me share the screen share screenshot.

 **Tarun Jain** 15:41

I just press this here.
So stream lit issues. What you can do is once you deploy stream lit right you will have logs.

 **Ayush Makwana** 16:03

Yeah.

 **Tarun Jain** 16:04

So let's suppose I click on this.

 **Ayush Makwana** 16:06

OK.

 **Tarun Jain** 16:09

On your bottom right side you will have manage app and then if you click on this three dotted icon you will have this download log. So if you have any issues with it you can just send me this download log this text file because this will help me to debug what is the issue.

 **Ayush Makwana** 16:28

OK, sure, I will send it.

 **Tarun Jain** 16:30

Yeah, and I'll mail this again back. Uh, this feedback, I mean.

 **Ayush Makwana** 16:38

OK, what?

 **Tarun Jain** 16:38

So this this is what you can change.

 **Ayush Makwana** 16:40

Hmm.

 **Tarun Jain** 16:43

So it's just like you have to run a loop and once you run a loop you will have this as a list. So this is not list, you'll have a dictionary. So this will be one key and this will be one key. And once you have the dictionary then what you are supposed to do is you have to change it here.

By running a loop.

 **Ayush Makwana** 17:00

Yeah, got it.

 **Tarun Jain** 17:04

And then you will have document which is in Lancen format which can later be used for chunking or even for evals.

 **Ayush Makwana** 17:13

Hmm.

Yeah.

 **Tarun Jain** 17:17

But.

So let's proceed. We'll start with Lama index today. I will share this collab notebook. So everyone understood right how evals works, how rag works, because that is the fundamentals.

Anyone has any doubts so far in Lan Chin?

 **Ayush Makwana** 17:43

But I love now.

 **Tarun Jain** 17:47

OK, so now why the what's the reason of using Langchain? I mean Lama index. So when we have Langchain, so when it comes to Lama index.

There are core functionalities which are currently missing in Lansing and when it comes to building your final application, I will always prefer Lansing. In Lansing also basically it is Langraph.

Right, but Lama index comes in if in case you want to develop certain POC which has to be very quick enough. So Lama index there are very limited lines of code you need to write and then you have your app ready. So if in case there is very urgent requirement and if you want to do multiple experimentation, let's.

Suppose you have re ranking right? You want to try cross encoders first.

And then uh, there was also something called as Flash Rerank.

And then you have coyer. So if you want to switch between this encoders, probably in Langsin you have to write around 4:00 to 5:00 lines of code. But when it comes to Lama index, you just have to write one line of code and similarly when we are creating answer node, right?



Tirth 18:57

Mm.



Tarun Jain 19:01

Answer not in the sense.

In line chain what you're supposed to do first you need to have retriever and once you have retriever you have augment and once you have the augment then you are running the generator.

So this is nothing but a chain.



Tirth 19:19

Mm.



Tarun Jain 19:19

This is a chain that you're creating in Langra for Langchain, but in Lama index this entire thing it is solved by just one single line of code which is called engine.

So if you want to do rapid experimentation of which technique you want to use, that is where Lama index comes in. But once you are aware OK, this technique is working fine, then what you can do is you can reuse the same components in langchain. So whatever code you have in Lama index, it is pretty much similar in langchain as well. So this is only for quick experimentation and POC purpose.



Tirth 19:55

OK.



Tarun Jain 19:56

So I'll back this. So one is POC relanking and there are two functionalities, two to three functionalities which is there in Lama index and that is not available in Lang chain is one is the fine tuning of embeddings.

So why do we need fine tune off embeddings? So let's suppose you have your data. Once you have your data, then you're chunking. And after chunking, what are you trying to do? You're saving it in a vector database. So let me open the slides.

So you have data, then you're have your chunking approach and then you have embeddings and once you have the embeddings you're saving it in a vector DB. So when you have your embeddings you're using the predefined 1. So in that predefined whatever data you have the tokens.



Tirth 20:44

Correct.



Tarun Jain 20:53

It might be there in the embedding, it might not be there in the embedding. So now what will happen is if you want to have your own language, let's suppose you have Gujarati language transcript. Now in the vector database that you're picking or the embedding model, if Gujarati keywords are not there, that means no information will be extracted.



Tirth 20:56

Mm.



Tarun Jain 21:12

And saved in the victory DB. So during that time what you can do is you can fine tune your embeddings. So once you fine tune your embeddings you will have your own embed model.

So this is one you have fine tuning of embeddings and second one we already saw.

Can anyone tell me what was that second feature?
We saw that code also.

 **Tirth** 21:48

Again, be just for what? Sorry.

 **Tarun Jain** 21:50

So there is one more few features which is available in Lama index but not in Langchain and we saw that code earlier the Lama index code.
The Triton MRR.
To evaluate.
Your embeddings.
And retriever.
You remember this iterate and MRR, so if I just.

 **Tirth** 22:25

Mm-hmm. When we were using S judge, right?

 **Tarun Jain** 22:30

No LLM as a judge is different. It trade MRR which is different.

 **Tirth** 22:32

No, no, when you uh sorry, when you said that uh script to uh you know to judge which embeddings or which model is good if you had that script.

 **Tarun Jain** 22:42

Yeah, this one.
Iterate an MRR. Here we are using Lama index.

 **Tirth** 22:44

Yeah.
Right, right.

 **Tarun Jain** 22:48

So in Lama index we had a game face and weddings and then we had simple

directory reader and then you had vector store index. So if you see here this line over here it is the only line you need to create your index and this itself is your engine also. So what line Lama index is doing is.

Whatever you see here, right this black cell, this part, this is just one line of code.



Tirth 23:11

Hmm hmm.

Hmm.



Tarun Jain 23:17

So you ask a query and once you ask a query it is converting into embedding, looking into Vector DB, getting the context, giving it to the LLM and then you have the response. So all these five step it is just one line of code and it is faster than line chain.

So.



Tirth 23:32

Why not use this then? Like in production you said that you would go for length and length graph.



Tarun Jain 23:38

So again, it's similar to the old Langchain. There are too many wrappers when it comes to Lama index and Langchain. So what Langchain did was they moved everything to Langraf and when they moved everything to Langraf, they made it modular. But these folks still didn't migrate.



Tirth 23:56

Sure.



Tarun Jain 23:58

So there are possibilities. You can combine LAMA index with Langchain also that is via this iterate and MRR. So this is 1 feature. Whatever is available in LAMA index, you can wrap that inside the Langchain component. But using LAMA index for production grade, it's not well suited, not many.

People are using it. There are very less case studies on Lama index.

 **Tirth** 24:25
OK. **Tarun Jain** 24:27

So this is 2 feature and 3rd and the most important thing is you have something called as Lama cloud which is nothing but Lama parts. So do you guys remember I told this Lama pass keyword earlier when we were discussing document extraction? So for document extraction we used something called as PDF. PDF from.

 **Tirth** 24:54
Hmm. **Tarun Jain** 24:55

And there is one more called as unstructured IO.

 **Tirth** 24:57

Unstructured correct and unstructured does everything, not doc file and everything right.

 **Tarun Jain** 25:00

So. Correct. So unstructured IO it's used for OCR and apart from OCR it also have their own chunking strategies.

And when do we use unstructured IO if in case latency is not an issue?

And PDFM. This is also very close to unstructured IO, but the OCR support is not that great.

OCR is not that great.

So when it comes to OCR, right, most of the models that you have, you have to host that model somewhere. You have libraries like Tesseract. So what this model does is you have to load that and these are very heavyweight models. So what Lama Index has done is they've created their own product and they kept it as Lama.

Cloud and in this Llama cloud they have a product called Llama Parts.

So llama parts what it will do is whenever you have tabular column, tabular data and

also OCR it will extract as it is.

As it is without any compromising of data. So you have PDFM, you have unstructured IO and then you have llama parse.

Did I mention that before Uber presentation?



Hardip Patel 26:37

Yes, you did.



Tarun Jain 26:40

So if you see it, you have llama parse, you have py PDFM and then you have unstructured IO. So these are the three libraries and then you have mark it down if you want to convert that into markdown.



Tirth 26:50

When when we don't know what kind of files will be coming through, should we always go for unstructured loader?



Tarun Jain 26:56

Unstructured loader is safest. If in case you need OCR you can use OCR. If in case you need a tabular extraction you can use tabular extraction. And when you are using unstructured IO you have to make sure there is a parameter called high res.

So this should be, uh, what you call.

Sorry, this is strategy. You have a parameter called strategy. That strategy should be high res. High res is nothing but high resolution.

So I resolution is good if you have OCR. If in case you don't have your data as OCR, then you can keep strategy to be fast.

Strategy unstructured.

So if you see one is fast and one more is iris. Iris is mainly when you have very sensitive data and you have OCR and then you also have OCR only. But if you use iris you don't have to pick OCR only.



Hardip Patel 27:56

Um.



Tirth 28:06

Understood.



Tarun Jain 28:08

And if you keep auto, it will select whichever strategy is required. It will based on your document data, it will pick automatically, but this will be slow.



Tirth 28:17

Mm.



Tarun Jain 28:18

So if you know your data already, then you can pick either Fast or Iris. These are the two best options.

And the parameter is strategy.



Tirth 28:34

OK.



Tarun Jain 28:37

And this is the slides probably I showed this earlier by PDFM and these are the three libraries, but this is paid like you have to pay for the API key.

So this is what LAMA index provides, which is their core what you call business. If not all these are open source framework. When it comes to Langchain, their main source of income is the Langsmith, which is their observability tool, which is again paid.

So instead of Langsmith, we used Ragas.

Ragas plus opic.

So Langsmith also provides what Opic does.



Tirth 29:24

That's good.



Tarun Jain 29:26

So let me take the screenshot of this.

Cool, is this clear why we are using LAMA index? This is only if in case you have very

urgent requirement and if you're trying to experiment with different reranking in Lang chain you have to change multiple code to do this, but in LAMA index it's just one line of code you have to change.

Now that you understood cross encoder is good enough, then what we can do? Run evals, migrate to langchain and migrating from llama index to langchain is very easy. And that you'll also see in the code because most of the syntax is very much similar. The only thing is functions will be different.

Oh, did I share the collab?



Ronak Makwana 30:21

Oh, yes.



Tarun Jain 30:22

OK, we can create the copy of this.

I'll start with the installation. So I hope this line is same. We know we are using vector store and then we have fast embed, fast embed and then you have LLM which is your Google Gen. AI. So key components that we need to understand here is.

If you want to upload your own data, what you can do is there is a folder called Simple Simple Directory Reader. Inside Simple Directory Reader you just have to create a folder. Whatever data you have, whether it is PDF document, CSV, CSV typically I will not refer to use.

For rag so docs, PPT and PDF you can use and then if you have YouTube you can have your different loaders and once this is done in llama index they already have their own chunking strategy. So chunk it's referred as nodes.

So chunk is referred as node.

And then if you want to save your data inside vector store, the concept is called storage context.

So storage context is nothing but you're configuring your vector database. If you're not configuring your own vector database, by default they have in memory vector store.

I hope everyone remembers what in memory is. It is creating a folder only within the project directory and it will vanish once you delete your data. And if you're using collab, once Internet goes off, your data is gone. So that is called in memory vector store. This is by default.

By default and if in case you want to change into your own vector store, what you

can do is you can define your quadrant vector store or you can define your milverse. Vector store or you can define Postgres.

Post SQL. So these are your vector store. You have to define our vector store and then use it in the storage context. One is node. After node you have storage context. Once you have storage context you have vector store index.

So vector store index is where you combine your data.

Embedding.

How that's it? Data and embedding and embedding is by default saved inside storage context, so embedding.

It already has embeddings. So here when you are defining vector stored index, you just have to define your data and storage context.

Is this clear? Node storage context for vector store and in storage context you have embeddings already defined, so by default it is again open AI.

Open AI is default.

And once you have storage context, you're creating your index. So index is nothing but you're saving your data inside vector store. Once you have this ready, the next thing is answer generation. So for our answer generation you just have to run one line of code which is engine.

Once you have engine, you can use it for retriever.

Or you can use it for query.

So 123 and four. These are the four fundamentals of Lama index.

Oh, is this clear?

Probably we'll see in the code it will be easier. So when we come to answer generation here also LLM is default.

LLM is default and that LLM is open AI. When it comes to langchain you have to define LLM and then you have to give it to your chain right there you are defining it. Here it is already defined which is default. If you want to change it you will use something called a settings.

Settings dot LLM equals to what LLM you need. I need Gemini which is Google Gemini and if you want to change the default embeddings you will do settings dot embed model.

Equals to fast number.

And if you want to change your chunking strategy, what are the two variables we use for chunk?

2 variables that we define for chunking.



Tirth 35:25

Then 10 parcel no.



Tarun Jain 35:28

A what?



Ajay Patel 35:28

That is for search, not ancient passes for search.



Tarun Jain 35:32

Denson sparse is for search.



Ajay Patel 35:35

For chunking size, not what size.



35:35

Oh.



Hardip Patel 35:39

Total characters to have an overlap, chunk size and overlap.



Tarun Jain 35:41

One is sun size.

And 2nd is settings dot some call up.



Tirth 35:45

Mhm.



Tarun Jain 35:53

So these are already defined. So when I said right when you're loading your data it already has chunk which is not. If you want to change that you can do settings dot chunk size in storage context also embedding is there but it is by default open AI. But if you want to change this, you have to do settings dot embed model and whatever model you want to use. Same goes for engine engine as LLM and if you

want to change that LLM to your own default you have to do settings dot LLM.

So this is the core fundamentals of llama index.

And once you have installed, you can probably click on restart session.

And this is again same code. We have to save our API key inside Google API key and once you have saved you can just save it inside environment and most of the operations that takes place in llama index is essential. So it's better to use these two lines of code which is import.

Next Asinsure asinsure dot apply.

And even for iterate and MRR, we had these two lines of code.

And now what are we trying to do before we proceed by adding our own data? Let me remove this.

Before we proceed to add our data inside chunking strategy, what are we supposed to do? We need to have our LLM as Google, Google Gemini and settings embedding model to be fast embed. So once we configure this then we will proceed with data loading, storage context and indexing.

So here the LLM what I want to use is Google Gen. AI which is from lamindex dot LLMS dot Google Gen. AI import Google Gen. AI and then for fast embed you have lamindex embeddings dot fast embed fast embed embeddings.

So the syntax is same, but the only thing is name is different.

And model is Gemini 2.5 flash max token 1024. If not you can keep this as none and temperature I'll keep it 0.1.

And now instead of invoke in Lama index you have LLM dot complete.

Similarly, if you saw our code yesterday, right? What function did we use?



Ajay Patel 38:36

Graph loading book.



Tarun Jain 38:38

A what?



Ajay Patel 38:39

Graph working working for this we have created this.



Tarun Jain 38:43

So for LLM, so LLM here is LLM dot complete and for line chain it is LLM dot invoke.



Tirth 38:53

Mm.



Tarun Jain 38:54

Then prompt. Then what are we supposed to use here?
dot content.



Tirth 39:01

Um.



Tarun Jain 39:02

So yesterday when we wrote our own functions, what did we define?
Inside scripts if you see you have Raga's line chain then you have client. So this is
how we define. So whenever we use LLM for our code we have to use a function
called get response.

So similarly in llama index it is called complete. Then here it is invoke. In our code it
was get response. So if you're building your own library, this is how you can define
anything whichever is naming convention. If it is relevant you can keep it.

If not, you can also keep run. Then what people will do? They just have to use LLM
dot run and the prompt.

Is this clear? So only functions will be different, but most of the core functionality is
same.

It's just the rappers.

So now embeddings equals to fast embed embeddings. You have model name. Here
you have Gina AI, Gina AI embeddings.

So when I meant trappers, at the end of the day when I'm using Lama index, if I'm
using Google Zen AI in the back end, you're using Google library only. Same thing
what even Langchen is using whenever it comes to fast embed embeddings in the
back end you're using fast embed.

The library is same. The only thing is the way you're defining it. The way you're
importing it is the only change, but the functionality is same. So I said you can work
with Lama index first. If not, you can directly jump to line chain if there is no urgent
requirement of any demo.

And now I just have to do from lamaindex.core import settings settings dot embed

embed model equals to embeddings settings dot LLM equals to LLM. That's it. So now if I just click on settings.

And if I just click dot and if I write C it should show chunk size and chunk overlap. So you can change chunk overlap, chunk size and can anyone tell me what we will use for callback manager?



Tirth 41:14

Opic.



Tarun Jain 41:14

Correct. So if you have OPIC and if you want to use OPIC with LAMA index, you have to define that inside callback manager which is settings dot callback manager.

I will delete this.

If not, let's try to use it because most of the code is already there. Let's try to use Opic here. So what we can do is add a new line.

Let me check the installation.

Lama and Doug fall back. Opic.

It installed OPIC.

Install. These are the two lines of code you need to run.



Tirth 42:46

Where did you got the callback one? Where did you got the callback opic from?



Tarun Jain 42:51

This one I used it from my previous code.

I don't know.



Tirth 42:59

OK.

It it's not mentioned here. Yes, it's not mentioned here.



Tarun Jain 43:10

OK, where did I get it from?

M.

Wait, probably last time when I log in here comment here they had integration.

So if I click Atyantik and if I click here somewhere you have integrations.

Oh.

Does anyone see integration keyword somewhere?



Tirth 44:26

And to look for it.



Tarun Jain 44:36

Not documentation. Somewhere here only in integrations we'll find it here. So click on this logger trees and then here you have multiple providers. You have Lama index. OK, here also they're using only open.

So I actually when I was working for this talk, I was using Lama index.

So during that time I had that code ready.



Tirth 45:07

Look.



Tarun Jain 45:07

But I don't know where I took it from now.

Lama Index.

High tier.

Love my index callback so pic.

It is inside integrations only integrations.

So I will clear this output and below this what I'll do is I'll just add OS dot environment OP KPI key.

Then OS dot environment opaque workspace.

OS dot.

Environment opaque.

Project name.

So same code cell that we have run. One is OPIC API key, then OPIC workspace and then OPIC project name.

So in our case, these two things will be different.

Is this done?



Tirth 46:56

And this is again.



Tarun Jain 47:09

By default callback manager is none, but for embedding model and LLM it is open AI.

So now what have we done? We just changed open AI to Gemini, then open AI to fast embed for embedding model and it's called as embed model.

The keyword and now for Opic, since it is a callback manager, by default it is none.

We want to append it as Opic.

So you're only import those two lines. Meanwhile, you can add these three environment variables.



Tirth 47:46

Correct.



Tarun Jain 47:47

One is from Lama Index.

dot 4 dot.

Callbacks.

I just have to import callback manager.

And then from opaque dot integrations.

dot.

Hot.

Yeah, these two lines. One is from Lama index dot core dot callbacks import callback manager, then from opic dot integrations dot Lama index import Lama index callback handler.

All these are outdated. We have to e-mail them.



Tirth 49:06

M.



Tarun Jain 49:07

Is it done? Here we added these three lines and then here we added two more lines.

And make sure you have installed it. One is OPIC and after OPIC llama index

callbacks OPIC.

This line.

And one downside of both the Lama index and Langchain both is the documentation is very bad. So if you have API references from API references you have to debug it.



Tirth 49:37

Mhm.



Tarun Jain 49:53

And for llama index also don't directly go with docs dot llama index. So recently they have made it as developers dot llama index. So here you have most of the API references and this is well maintained if you search for developers dot llama index dot AI.

Instead of docs dot Lama index dot AI. So this is the Lama cloud which is their main product. But if you are using unstructured IO you don't have to switch to this and framework is what we are utilizing and similar similar to Langraff even they have workflows.

So workflows is like Landgraf. So framework is like Linxay, Lama index, Langraf, Lama workflow, Lama cloud, Langsmith. So if you pick one library, second library by default you can manage.

So it's called developers dot lamindex dot El.

Uh, is it done?

Visual lines of.

Uh, hello.



Ayush Makwana 51:17

Yeah, it is done.



Tarun Jain 51:19

Cool. I'll also copy it if in case there is any spelling mistake.

So now what we need to do is when you come back to your settings dot configuration, here you just have to define settings dot callback manager. Whatever you have imported here, callback manager, copy this, paste it here and inside this what you need to do is create a list.

And inside this list, just copy this LAM index callback handler.

And this is a function.

So settings dot callback manager. By default it is none. Now what are we doing? We are importing callback manager. Inside this you need to have a list. So inside that list you're defining your Opic handler. So this Lama index callback handler it's coming from Opic.

And now what you can do is you can create a folder here.

Call data.

There are two ways you can do it. First way is if you want to upload an individual file, you can do the individual file. The second approach is you create a new folder, name it as data or it can be anything. I can also name it as documents so this can be anything.

Data.

And inside this I'll upload a PDF file.

So one file I have it inside data and one more what I will do is I'll upload a file which is outside.

Wait, this is some confidential data.

I'll upload the same document outside.

Oh, is it clear? One what you can do is create a folder inside the data file. One more what you can do is you can directly upload as an individual file.

And once this is done, just do from llamaindex.core import a simple directory reader and here I mentioned both the syntax. If in case you have created a folder and inside that folder if you have added all the files, what you can do is you can uncomment this.

You can uncomment this and then run this particular cell. But if you want to add only individual file then you can define it as input files, define it inside a list and then give it a path.

And I hope you know what is recursive equals to true.



Tirth 54:20

Yes.



Tarun Jain 54:21

So this is chunking.

So now I'll just comment this and I will run it.

And now if I do length of documents it is 6. In your case it might be different. So it is

already chunking the information and if you run documents of 0 you will see too much of information.

So here if you see you have ID but the format is same. So what was the data type used in line chain?

It was document. Inside document you had page content and metadata. Here if you see you have ID. Inside ID you also have embedding. Then you have metadata. So if I click on dot metadata you have page label, file name, file path, file type, file size, creation date and last modified.

 **Tirth** 55:07

Correct.

 **Tarun Jain** 55:23

The only thing is instead of source you have file name. In Langston it is source, so it's just variable changes and here instead of page content you have text.

 **Tirth** 55:36

Just just confirming this is only doing for PDF files or if I give MD files it would work as well.

 **Tarun Jain** 55:36

Text is where you have.

Yeah, it will work for the MD files as well.

 **Tirth** 55:45

OK.

 **Tarun Jain** 55:47

So here you can give any files. You can upload PDF, TXTMD docs.

 **Tirth** 55:53

So for Lang chain, do we not have anything similar as this or there is like simple directory reader?

 **Tarun Jain** 55:57

There is also there for uh you have directory loader.

Lunching, but here it has to be specific. If you're using PDF, it should be PDF only.

 **Tirth** 56:05

But.

OK.

 **Tarun Jain** 56:15

Oh, how did we define five PDFM?

 **Tirth** 56:21

We gave path to the PDF files with path equal to.

 **Tarun Jain** 56:24

Offline document processing Uber.

 **Tirth** 56:33

Because I did a bit of over engineering for all the different kind of files. So you know, we used to, yeah.

 **Tarun Jain** 56:36

So you have five PDF. So by PDF you have something called as generic loader. So if you have multiple files inside a folder, let's suppose I have a data. So what will I do? I'll define this file system BLOB loader.

 **Tirth** 56:46

Hmm.

 **Tarun Jain** 56:52

Inside that I have path. Path is nothing but the directory. Inside that I'm only considering PDF and for this PDF what loader am I supposed to pick by PDF from 2 loader? If you're using unstructured IO, you can define unstructured.

 **Tirth** 56:58

Hmm.

That's sure.

 **Tarun Jain** 57:08

So here if you just open loaders, you have multiple loaders, document loaders.

 **Tirth** 57:13

If we go with unstructured, if we go with unstructured then it will do all the files now.

 **Tarun Jain** 57:18

No unstructured or you you are picking right in unstructured also it's separate. So let me open this. Can you see this documentation on the left hand side?

 **Tirth** 57:29

Yes, yes.

 **Tarun Jain** 57:31

OK, I'm just scrolling down so if you see unstructure.

 **Tirth** 57:34

Why? Why I'm asking this? Because right now the project that I'm working on, it has a mix of lot of files including images and everything because of the screenshot that we took randomly and everything right? And then I had to define for each and every file type what kind of loader I'm going to use for chunking and document generation.

 **Tarun Jain** 57:52

OK, you want data type. I mean just one loader which will take multiple files.

 **Tirth** 57:53

And I.

Yes, like this one. Like this is pretty good from index one, right? Like if you just give the directory and I'll just give you the documents back.

 **Tarun Jain** 58:05

So unstructured Word document is there. Then this is for XML.

I don't think they have it for multiple files.

Right, so this is for PDF constructed PDF product.

9/10.

But you'll have to explicitly define it. If it is web URL, you have to define it like a web URL. It's not like Lama indexing.



Tirth 58:49

Mhm.



Tarun Jain 58:53

So here if you see the this thing will change. If your data is path, it will be like path.



Tirth 58:55

Yeah.

8.

Hmm.

Mm-hmm. Correct. I think that is what I did. But can we not use this mix like you know the documents we expect from Lama index into Lang chain, no.



Tarun Jain 59:11

So what you can do here is.



Tirth 59:13

Hmm.



Tarun Jain 59:15

You have simple directory reader. You're extracting all the information. Now what I show to Ronak, right? Sorry, not Ronak Ayush. Where is the document?



Tirth 59:17

Mm mm.

Hmm.

 **Tarun Jain** 59:32

So this will be the logic for document in documents, which is this one.

 **Tirth** 59:37

Mm mm.

 **Tarun Jain** 59:39

I have this defined above.

 **Tirth** 59:42

Hmm.

 **Tarun Jain** 59:43

And if I copy this.

 **Tirth** 59:47

Alright, OK, so we're now converting it to Lincoln document.

 **Tarun Jain** 59:50

So now this will be like don't keep it this as document, I'll keep it as docs.

 **Tirth** 59:55

Hmm.

 **Tarun Jain** 59:57

Docs equals to document. This page content will be.

 **Tirth** 59:59

Um.

 **Tarun Jain** 1:00:02

Document dot text.

 **Tirth** 1:00:03

Or text.

Oh.

TJ

Tarun Jain 1:00:09

And this metadata will be document.



Tirth 1:00:12

OK.

Understood.

TJ

Tarun Jain 1:00:17

Document dot not document. Here document is already OK, this is not defined document.

But let me keep it different because this document is there. I'll keep this as just DOCR info.



Tirth 1:00:31

E.

TJ

Tarun Jain 1:00:35

4.

Info dot text info dot metadata.



Mitesh Rathod 1:00:41

OK.

TJ

Tarun Jain 1:00:41

That's it.

And all this thing needs to be appended. So not appended you can use extended. I'll keep data equals to list.



Tirth 1:00:47

Yeah, certainly.

TJ

Tarun Jain 1:00:57

Because this is individual component now, so you can either docs dot extended. I usually use extended then you can add docs, sorry dot extended docs. So now if I print blank chain.

 **Tirth** 1:01:00

Mm-hmm.

M.

 **TJ** **Tarun Jain** 1:01:19

So Ayush you got it. You have to do the same thing. Here I already have documents, but in your case it won't be this much straightforward.

 **Tirth** 1:01:22

Open.

 1:01:24

Yeah.

 **Ayush Makwana** 1:01:28

Mhm.

 **TJ** **Tarun Jain** 1:01:29

But the syntax is same.

 **Tirth** 1:01:34

Although the the code that you have shared for recursive true, it is not going inside directory. So if I have multiple directories.

 **Ayush Makwana** 1:01:35

Yeah, I got it.

 **Tirth** 1:01:43

Or OK, it's still uploading. Sorry, my bad. Please go ahead. Please continue. Sorry.

 **TJ** **Tarun Jain** 1:01:48

So I will run this and now if I run this, now the data type of data is a line chain now line chain dot.

 **Tirth** 1:01:50

Yeah.

 **Tarun Jain** 1:02:02

Right.

Type of data it is list now data of zeroth index.

What mistake did I do?

OK, wait if I do open.

Now it's correct. So now data of zeroth index will have document dot metadata.

Have this metadata then data of 0 dot page content.

Now this you can use in line once you have it in document format. Same thing you do for Lama parse. So we do have Lama parse subscription. So what we do is Lama parse is faster than unstructured IO. So you define what you call from Lama parse.

 **Tirth** 1:02:59

M.

 **Tarun Jain** 1:03:02

Once you have llama parse, the format is same. You will have metadata, you will have text that you can convert into into land chain. Since this is happening offline, you don't have to care about it, right? Like how much wrappers you're using. Once you have it in saved in Vector DB then the real.

 **Tirth** 1:03:09

OK.

 **Tarun Jain** 1:03:18

Issue starts because this you can run anywhere. Once it is in vector store then you have to see how you can improvise the performance.

 **Tirth** 1:03:19

M.

TJ

Tarun Jain 1:03:30

So I'll just copy this and send it.

So this is for everyone like if in case you have your own data which might be very rare case like the rare case was this one.

The transcript one. So if you saw the transcript, how it was like you had name, you had timestamp. So timestamp is very important. Let's suppose at one specific timestamp I spoke something and that is your query like hey, what did Tarun say at 15th minute?

And now if you need the response, what will you do? You will use filtering. So in that filtering, what is your key now? What is the key?

So what key should I use for filtering?



Tirth 1:04:21

In metadata, there should be timestamp. Yeah, timestamp.

TJ

Tarun Jain 1:04:22

Which is a timestamp. So now what will happen if I say, hey, what did Tarun say in the 15th minute? You will get that particular 11 context. So this becomes very good use case.



Ayush Makwana 1:04:24

Oh, I.

TJ

Tarun Jain 1:04:35

So, but the only thing is defining metadata and page content, you have to do little bit of reprocessing which is running for loops, saving it in the dictionary. So here it was straightforward. Why? Because we had info dot text and info dot metadata from Lama index.

So you were able to do it, but metadata is something that you have to customly create here in this use case.

Oh, is this clear the logic we're using this?



Tirth 1:04:59

Yeah.

Yes.



Ayush Makwana 1:05:02

Yeah.



Tarun Jain 1:05:03

OK, so now what we have done is we completed this thing. You have your external data, chunking is done. Now as I mentioned, you have something called a storage context which will take care of embeddings by default. So you don't have to explicitly define it and we have already appended it in the settings.

Now the only thing what you need to do is define your storage context. So when you're defining storage context, what you're supposed to do is here if you see storage context equals to storage context from defaults, this will be empty.

If this is empty, what is it using?

So if this is empty, what kind of vector store is this?



Tirth 1:05:51

Dense vectors.



Tarun Jain 1:05:53

No, this is in memory vector store.



Tirth 1:05:56

OK, OK.



Tarun Jain 1:05:57

So this is in memory vector store. If you don't have any vector database and if you still want to build a rag up, you can do this. But this is saved in your in memory vector store. And what kind of indexing does it use? The indexing algorithm is HNSW.

Which is, uh, Iyer RQL small world.

Iyer RQL.

Small world. Did I use this word before when I mentioned the quadrant in line chain?



Tirth 1:06:35

You did. You did.



Tarun Jain 1:06:36

So this was like how do you connect one person to different person and the layer?



Tirth 1:06:41

Right.



Tarun Jain 1:06:42

So you had M and you had candidates.

So this is by default. This core algorithm is used in in memory vector store, but if you want to have additional functionality, that is when you define your own vector store.

So here you can just define vector store.

Equals to vector store.

So if you see you have something called as base pydantic vector store. This is nothing but your in memory vector stored. Is this clear? And this logic is you already know. So this is in memory vector. What it will do it will create a folder here quadrant.



Tirth 1:07:19

It's.

Yes.



Tarun Jain 1:07:27

And then you are defining the collection name and once you define the collection name, what is the next step? You need to create collection. So collection equals to collection. Then you have vectors config. So vector config what is the size 768 and what is the distance cosine.

So this three cell are same. Once you have this three cell you have quadrant vector store. The name is same. Even in line chain the name was quadrant vector store. Inside that you have to define only client and collection. Why are we not defining embeddings?

Because embeddings is already added here.

Is this clear?

In line chain we had three variables. In quadrant it is only two variables because embed model is already inside settings.

So this is the new line we are writing. This was same as per line chain. Now you just have to define storage context from defaults is the function then append your vector stored.

So now we completed till here. Now once you create I mean once you save your embeddings inside vector store you have to create indexing. So indexing will have two things. One is you have to give embeddings and you have to give your data which is the CNC.

So for that you have something called as vector store index. Define vector store index from documents. Why documents? Because the data type of this is documents. If I do documents, if you see the zeroth index, this is document. If it is text. If you have raw text then it will be vector store from.

Vector store from document and if you have already saved it, what you can do is vector store dot. You have something called as a vector store. This is after you have saved it, but as of now we have not saved it yet. So what are we supposed to do from documents?

This will take some seconds.

Like 10 to 20 seconds depending on your data and you should see this warning message, not warning message, logging message from Opic. So now whatever you ask any question, it will be traced inside this particular URL.

So if you see index construction was done.

And you had chunking one, chunking two, how much time it took for every single chunk. It has the time and then the embedding size and what is there in each chunk you have the information. This is what indexing is index construction.

This was not there in Lancenagers.



Tirth 1:10:15

Something didn't work for me. It is saying that lemma index callback handler dot start trace missing one parameter self.



Tarun Jain 1:10:24

Uh, did you add it as a list?



Tirth 1:10:27

Uh, checking. Yes, callback handler and then list and then list inside. Llama index callback. Oh, I'm not calling it as a function. Uh, sorry, my bad.



Tarun Jain 1:10:29

So this should be.



Tirth 1:10:37

Sorry, my back.



Tarun Jain 1:10:38

We did this mistake yesterday when we define Graph Builder.



Tirth 1:10:39

OK.

Mhm.

OK.



Tarun Jain 1:10:45

So when we wrote this main function here graph builder, what we did was we did graph builder dot you know we didn't write function. This was missing.



Tirth 1:10:52

Mhm, OK.



Tarun Jain 1:10:54

So today also this this function.

So is the index created? Are you able to see these logs everyone?

Let me know how many of you have completed.



Tirth 1:11:13

It's still getting this llama index callback handler has no attribute even starts to ignore.

See what I did wrong.

So.

TJ

Tarun Jain 1:11:25

Oh, can you cross your uh?



Tirth 1:11:27

Yeah, I'm I'm checking.

Let my index of the integrations and contact minutes of.

The thing.

TJ

Tarun Jain 1:12:08

You got to care.

So how many of you are able to see this on Comet?



Ayush Makwana 1:12:21

My my collab session just restarted. I'm working on it.



Tirth 1:12:28

I'm getting a little. Let me shut my screen. Sorry.

TJ

Tarun Jain 1:12:32

Oh, is your document very big? Then what binary quantization?



Tirth 1:12:36

My document is very big, yes.

TJ

Tarun Jain 1:12:40

Here we can add binary quantization.



Tirth 1:12:43

OK.

TJ

Tarun Jain 1:12:46

So I'll just search for vector database.



Tirth 1:12:58

I have 428 documents.



Tarun Jain 1:13:03

It will take time.

So we can add this line binary quantization.

Quantization config equals to models dot binary quantization, then binary equals to models dot binary quantization. Always time should be false. If you keep always time to be true, it will hit the collab RAM.

This is to those who have very large uh data.

So now what we'll do is we'll go with answer generation. In answer generation, as you see, we just have to run one single line and that is called index dot as query engine. Once you have this as query engine, you can directly use query engine dot query and ask your prompt.

So here I'll ask what is this data?

Breakdown these steps regarding.

So did we do augmentation?



Ajay Patel 1:14:47

Mm.



Tarun Jain 1:14:47

So if you see after indexing, usually what are you supposed to do after indexing? You need to define retriever, you need to define augmentation and then you have to have generation. All these three things is already taken care by llama index and they have very solid prompt. So if you want to check the prompt for this, what you can do is create a new code.



Tirth 1:15:08

I'm still getting errors. Sorry, can you help me with that?



Tarun Jain 1:15:09

But.

Yeah, uh, can you share your screen?



Tirth 1:15:13

Yeah.

OK, I'm getting this error type Lama index callback handler no attribute event starts to ignore.



Tarun Jain 1:15:29

OK, can you scroll up?



Tirth 1:15:31

Yeah.

I did the binary configuration.



Tarun Jain 1:15:37

This is coming from Callback Manager uh settings. Can you scroll up?



Tirth 1:15:44

Callback manager.



Tarun Jain 1:15:45

One second callback manager, Rama index callback handler. Can you check the import statements once?



Tirth 1:15:53

Yeah.



Tarun Jain 1:15:56

We can do this now.

Uh.

One second.



Tirth 1:16:05

I'm just checking here. It is there. Typhon. OK, yeah.



Tarun Jain 1:16:08

No, their documentation is gone, messed up. Oh, can you click on runtime, restart session and run again once?

 **Tirth** 1:16:15

Good.

 **Tarun Jain** 1:16:16

Yeah, restart session.

 **Tirth** 1:16:19

OK, I'll.

 **Tarun Jain** 1:16:20

Uh, don't run with the library. Uh, can you scroll up?

 **Tirth** 1:16:23

OK.

 **Tarun Jain** 1:16:24

I start from the third cell.

One second OP KPI key Opic workspace Opic project name OK.

 **Tirth** 1:16:40

Yep.

 **Tarun Jain** 1:16:41

OK, integration start. Just one second.

 **Tirth** 1:16:46

Yeah.

 **Tarun Jain** 1:16:48

OK, or.

dot dot.

OK, that's correct. Opic dot integrations for long index. Yeah, how many of you are

getting the same error? What is getting?

Yeah, you can run it.



Tirth 1:17:10

OK.



Ishan Chavda 1:17:11

I I the same error but I restart the session and it is working now.



Tarun Jain 1:17:17

Oh, what happened?



Ishan Chavda 1:17:18

I got the error thing but I have restart this session and it is working fine for me.



Tirth 1:17:24

OK, so restarting session worked fine for you. OK.



Tarun Jain 1:17:28

Or maybe because once we run OPIC and OPIC integration, we didn't restart the session.



Ishan Chavda 1:17:29

Yeah.



Tirth 1:17:35

OK.

Reminder.

Um.

It might already exist. I just did it before.



Ishan Chavda 1:17:44

Oh.

No, you'll have to update the collection name.



Tirth 1:17:51

No, no, but I'm changing the DB now altogether.



Ishan Chavda 1:17:55

OK.



Tirth 1:17:59

OK, now it started. OK, it was just stationary start. Thank you.



Tarun Jain 1:18:03

OK.

But this won't happen if you run on VS code.



Tirth 1:18:08

OK.



Tarun Jain 1:18:10

Yeah, I'll share my screen again. Uh, the last part is missing.



Tirth 1:18:14

OK.



Tarun Jain 1:18:15

Uh, so.

Everyone got the response. So now if you see the code after indexing, we directly jumped into answer generation by skipping retriever and given augmentation and generation. Lama index already has a very good prompt and we can use the base prompt itself.

And one additional thing what lament X does is after you have prompt it also has something called as synthesizer.



Tirth 1:18:46

Sure.

 **Tarun Jain** 1:18:46

So I'll come back here.

This is not two or three.

This is 4.

So you're running two prompts with one LLM call.

So one is your augmentation prompt and one more is synthesizer prompt. So this synthesizer prompt is already by default inside your engine. So what you have to do is just define your query engine dot get prompts.

So if you see you have something called as response synthesizer and you have text QA template which is mainly used for rag and inside that if you see you have something called as template. This is what I needed. So if I want to get this outside.

Let me check the syntax.

 **Tirth** 1:19:43

To take care of.

Augmentation part. Can we do the same for identity classes over here?

 **Tarun Jain** 1:19:53

Oh, for what?

 **Tirth** 1:19:55

Like a structured output.

 **Tarun Jain** 1:19:56

Yeah, yeah, structure output is also there. You can define. So if you see output parser is there, which is none. If you define output parser, you'll have it here as spied antic.

 **Tirth** 1:20:00

That's output version. Yeah, yeah, OK.

OK.

 **Tarun Jain** 1:20:07

So this is dictionary. I will copy this response synthesizer.

OK, why is it an error?

Response synthesizer.

OK, response synthesizer colon.



Tirth 1:20:30

Take you with a plate.



Tarun Jain 1:20:37

OK, now what inside this?

Inside this.

AAP template Varams.

Then you have template.

This template is inside default template dot default template.

Now you have dot template so this is a prompt if I do print.

So you have context information is below and this is your. Where is this coming from? This is coming from retriever and I hope this syntax is comfortable to everyone. Wherever you see curly bracket, this is called input variable. Given the context information and not prior knowledge, answer the query. That's it.



Tirth 1:21:19

Mm.

Wait.



Tarun Jain 1:21:28

And then they have one more prompt and this prompt is response synthesizer refined template.

So the original query is as follows. Query STR. We have provided an existing answer. We have the opportunity to refine the existing answer only if needed with the more context below. All this thing it is taken care by Laman index.



Tirth 1:21:57

So.

2.

Good.



Tarun Jain 1:22:00

So for quick experimentation you can use it, but I faced multiple issues when I deployed it.

 **Tirth** 1:22:07

OK, OK.

 **Tarun Jain** 1:22:08

So I'm not sure with the workflows part.

But this is mainly for agents. For rag you still use lemma index only and in all the documentation you have same code everywhere. That code is they won't use any vector store here. They're using in memory and directly they're using simple directory reader.

 **Tirth** 1:22:18

M.

M.

 **Tarun Jain** 1:22:31

But we usually use llama parse here because llama parse is good and we have some free credits and then load it in langchain.

So now what if you want to change your prompts here? If you see this is a basic template and I don't need this template. So if you want to change this template you can do it. Why? What is the format of this? What is this data type?

 **Tirth** 1:22:58

Dictionary.

 **Tarun Jain** 1:22:59

It's a dictionary and it is mutable. So now what I'll do is whenever you're defining your own template, own prompt, make sure you have two input variables and those two input variables should be same context, STR, not context.

 **Tirth** 1:23:02

Hmm.

TJ

Tarun Jain 1:23:15

So it should be same and then query STR.

So if you need that input variables, you can just copy this template params.

Response.

Synthesizer.

Text QA template.

Then just print this template params.

Order.

And you have these two input variables, context STR and query STR. This has to be in your own prompt. Now what I'll do is you should.

Answer as an assistant working as an.

Error handling.

Chatbot.

You only have a contest.

If the query is not in the context.

Just say I don't know.

And this is the context.

And I will close it inside XML.

And then query.

Reading.

Till here everyone are following it. What I'm trying to do is if you got the output as response dot response, we are checking the default template because we skipped the augmentation and if you want to change and keep your own prompts, what you can do is.

You have this dictionary response synthesizer text to a template. Inside that you have default template and temp template. What I need to do is I need to append this. So for that now I'm writing my own prompt. When I'm defining my own prompt, I have to ensure that the input variables are same.

Which is context STR and query STR and once this is done you just have to define your query engine.

dot you have something called as update prompts.

Inside update prompt, whatever your dictionary is there, define the dictionary. So what is the key? This is the key.

And here you just have to define your own prompt.

That's it.

So now if I do query dot get prompts.

And now if I check this one.

You have your own prompt.

And once you save it, right, let's suppose once your data is saved inside vector store index, you don't have to define your documents. And if you're doing inference in a different file, if you're doing inference on a different file, how should you define it?

Again, define your vector store index.

Which has a client which has a collection and instead of from documents you will use from vector store which is already saved. So now if you see here I'm not defining documents so this will happen in app dot py.

So you define your vectors for index.

With same client and same collection name and then define your vector store, then define your query and define the response. So if you see this is very fast.

And then you have a response.

And you can also open Opic and if you see everything is traced.

Your retriever how much time it took? Your embedding how much time, then how much time the synthesizer is taking. So if you see this is where your refinement and prompt is using and then you have LLM. So if you are using prompt, obviously this time is for the LLM which is 2.3 seconds.



Tirth 1:27:55

It reduces a lot of thing that we do manually with line chain.

Like there might be issues, but it does reduce a lot of thing, you know, I'm just looking from that perspective.



Tarun Jain 1:28:00

Correct, Sir.

Yeah, this is very fast in the sense if you have very urgent requirement and if you want to do any demo right in 20 minutes you can build it in llama index. Once you build it right you just have to have app dot PY then use any copilot or cloud tell it to generate a streamlit app and you have your.



Tirth 1:28:10

Mm.

Hmm.

 **Tarun Jain** 1:28:25

Enter app ready in 20 minutes, 20 to 30 minutes.

But it's very good for only POC. After POC, once you start getting real time feedback, you will start feeling like OK, we have to migrate.

 **Tirth** 1:28:40

Thank you.

 **Tarun Jain** 1:28:42

But yeah, this is very good for POC purpose.

So we'll also see one poor example of how to fine tune embeddings.

And uh, hybrid search. I'll show an example.

With reranking. So this two examples I'll show, then we'll move to agents.

So I wanted to give one task because tomorrow I won't be available. So 26 to October 1st I'm again traveling to GD summit. So during this time what we can do is now that you have this code.

Yeah, I.

So if you remember in transcript you have ragas right and most of you are still not that much confidence with evals. So what you guys can do is refer this evals script and also the notebook you have traditional rag approach.

So most of the code that I've seen, whether it is food recipe or even transcript, either you're using traditional drag or you're using hybrid search, correct? So what you can do is create 2 files, one app dot PY has traditional drag, one more has hybrid search.

And one more what you can do is you can have MMR plus.

Re-ranking. So what is the evals for every approach that you are taking? Now after all these three approaches, whichever has more evals, that is our final project, that is our final approach.

Is this clear? So now let's suppose in this particular code Ayusha used hybrid search. OK, so here you have hybrid search. Did you use MMR?

 **Ayush Makwana** 1:30:46

Yes, I did.

 **Tarun Jain** 1:30:51

So it should be nav dot PY.

Point struct. OK, you're using MMR. How much diversity are you using?

0.3 OK not you can keep it 0.5. So now if you see you have used MMR with hybrid search. So this was your approach right?

Now what you can do is you can run event. You can get the score. You can also try re-ranking with MMR and then you can check the scores. So for re-ranking I'll just recommend you score here because it is fast and one more is traditional drag.

Traditional drag in the sense don't use hybrid search.

 **Ayush Makwana** 1:31:21

Yes.

 **Tarun Jain** 1:31:40

Don't use this sparse model, just use dense.

 **Ayush Makwana** 1:31:44

OK, yeah.

 **Tarun Jain** 1:31:46

And what you can do is you can have the scores of all these three. Whichever approach has more score, that is your final code. That is your final approach which you can deploy on streamlet.

 **Ayush Makwana** 1:31:59

Hmm.

 **Tarun Jain** 1:32:02

And if you notice main dot PY, this is where you'll have your stream lead code. And if anyone wants to go one step ahead, what you guys can do is you can build a discards at what.

 **Ayush Makwana** 1:32:02

Yeah.

 **Tarun Jain** 1:32:16

Or if you're using Slack, you can build a Slack chatbot instead of stream it. And where this logic will go, this logic will go in main dot PY. Your remaining code remains the same, client dot PY same, your state dot PY same and even your search technique which is your indexing that is same.

The only thing what you need to do is where will the user ask a question and where will he see the output for which we use Streamlet. But if anyone is interested and if you want to build a full-fledged app, you can either go with Discord chatbot or Slack chatbot. So when you go with this approach, make sure you decide which.

Roach is best.

Among these three, whichever has more events, then you can update your main dot PY streamlet or discord chatbot or slack chatbot.

And I guess copilot will give you the code for this.

This is something that no one has to worry about. The only thing is fundamental understanding. And if you know where your app is breaking and if you can debug it, probably that was the success of RAG. This is something that anyone can build.

So is this clear OK?

 **Ayush Makwana** 1:33:31

M.

 **Tarun Jain** 1:33:32

So with this we have complete track. The only thing is these are two functionality. Again this is optional but it is not much of the code in llama index. Hardly this is just three lines of code and even this is 2 to 3 lines of code.

 **Ayush Makwana** 1:33:34

Yeah, yes.

 **Tarun Jain** 1:33:49

So I'll show this on October 2nd and then fast API.

Plus.

SQL This I'll complete on October 2nd.

Basically the last virtual session. Then on October 3rd we can have agents, MCP and memory in person itself.

So these are the major things and I hope you have this results.



Tirth 1:34:23

I don't know this e-mail ID that you're talking right now. Can I share the GitHub repo as well so that you can, you know, just go through it and see what I'm doing wrong with, OK.



Tarun Jain 1:34:30

Yeah, I'll what I'll do is I'll usually check the code and then I'll share this as a feedback.



Tirth 1:34:35

M.

Yeah, absolutely, absolutely.



Tarun Jain 1:34:46

OK, yeah, that's it. But let's make sure you have 3 approaches, Events, because if you just have two approaches, that will not make much sense. Either this can be hybrid search and this can be MMR plus re-ranking. If not, you can have MMR plus hybrid search.



Tirth 1:34:48

OK.



Tarun Jain 1:35:04

Plus re-ranking and this can be MMR plus hybrid search. So we can have this three what you call this three combination.

And I believe this approach will have more scores.

Because here re-ranking again Hoyer will might have some limitation when you deploy and this is again API based. So after some limit you'll have to pay the price for it, but this one I don't think you'll have to pay for it.

Unless you're using Open Yard.

Because again, Gemini also has a limit and you guys can use Azure Open AI. This is the best and you have free credits for Azure Open AI.

 **Tirth** 1:35:53

Hmm.

 **Tarun Jain** 1:35:58

Where did that go?

Yeah.

I'll add all these things in the slides so that you can refer to it.

Oh, any questions before we wind up?

 **Tirth** 1:36:25

OK, all good.

 **Tarun Jain** 1:36:27

OK, I've added in the slides the reason of using Langchain and also the steps for.

Using Azure Open AI.

 **Tirth** 1:36:42

The.

Thank you all.

 **Ajay Patel** 1:36:52

Thank you. Thank you.

 **Tarun Jain** 1:36:52

Yeah. Thank you, everyone.

 **RamKrishna Bhatt** 1:36:54

Thank you all.

 **Ajay Patel** stopped transcription