Python and AI Power-Up Program Offline Class-20250901_113242-Meeting Recording

September 1, 2025, 6:02AM

1h 44m 17s

Ajay Patel started transcription

Tarun Jain 0:04
ESV file is about and once you explore it, most of the commands are same. So do we have any questions in the those 3 frameworks? Pandas, Matplotlib and numpy?

Tirth 0:15
Not as of now, but Matlab Matplot looks very promising for image generation of data.

Tarun Jain 0:23
Right, right for creating graphs.

Tirth 0:24
Yeah, the similar project in JavaScript, yeah, yeah.

Tarun Jain 0:30
So if anyone has to build any sales agent right during that time, that library is very very helpful. Anything that is related to business side?
OK, so today what we'll do is we'll start off with RAG and there are two popular frameworks that we have in RAG. One is Line Chain and Lama Index. So once we understand what are the important components of Line Chain and Lama Index, we'll also try to build our own RAG application.
And once you understand how to build the basic RAG pipeline, we will look at some of the case studies on how people are improvising this pipeline to get better results. So that is where we have advanced RAG as well. And how do we get to know which RAG pipeline is working well, whether it is the traditional one or advanced one, there are different techniques.

Tirth 1:11

Mhm.

Tarun Jain 1:19

You need to have certain benchmarking or evals, right? So how do we build evals is the next part after we explore that. So let's begin with what is RAG. In simple terms, if you want to build chatbot like ChatGPT for your own data, this is where you use RAG. So let's suppose you have user.

Tirth 1:29

Mhm.

Tarun Jain 1:39

User query and this user query is what is the work experience of Tarunjit. So if I ask this prompt to large language model under 1% it will give me wrong response only to not wrong response but it will be like it will give the response.
By the name Tarunjan who is famous. It won't be specific to me right? So what you will do is you will write a your user query, convert that into embedding model so everyone understood what is embedding model.

Tirth 2:00

Mm.

Tarun Jain 2:11

So we looked into fast embed rate, the vector search one. So in order to generate vectors we will use embedding model and then it will look into vector DB which has cosine similarity as the distance measurement and based on the search technique it will return the relevant context.

Tirth 2:15

M.

Tarun Jain 2:28

So now let's just imagine in this vector database I have my CV saved. So now if I ask

what is the work experience of Tarun Jain, it will look into the vector database and retrieve the relevant chunks right and once it retrieve the context. Now if you notice this LLM part.
Is getting 2 inputs. So what kind of prompting is this?

Tirth 2:50
M.
You're getting the context and you're getting the query.

Hardip Patel 2:56
OK.

Tarun Jain 2:58
Uh, So what do we call this kind of prompting?

Hardip Patel 2:59
Yeah.

Tirth 3:01
OK.

Tarun Jain 3:06
No system prompt is like common for every LLM. So here it's mainly few short prompt. Few short is you have user query, you have additional context. You can also add some examples and examples is nothing but metadata.

Hardip Patel 3:11
OK.

Tarun Jain 3:22
So if you have context with metadata where you have examples and then user query, this is nothing but few short and then you have the final response.
And now the major question is how do we save our data into vector database and why is it retrieving chunks instead of the entire data that you save, right? So in order to understand this, let's imagine we have an external data.

So this external data can be Azure block, it can be your AWS S3 bucket, it can be GCP, Google Docs, it can be SharePoint and there are people who also uses Confluence right from this JINA board. So your external data can.

Tirth 4:03
Hello.

Tarun Jain 4:05
Or what happened?
Hello.

Hardip Patel 4:11
Um.

Tarun Jain 4:12
OK, so this external data can be anything, whether it is PDF, CSV, CSV is typically not supposed to be used if you're building RAG. So you have PDF, docs and PPT. The next step is you need to convert that into chunks.
So converting your external data into chunks is very important. Why? There are two reasons. So the first reason is large language models have limited context window.
So if you look at ChatGPT it has the context window of 200K then.
If you look at Gemini, it has 2 million and Claude recently made their context window as 1,000,000. But the issue with all this is there is a test called needle in the haystack.

Tirth 4:58
M.

Tarun Jain 4:59
So in this needle in the H stack, if you notice most of the LLMS despite having 200K after certain point it starts losing the information. So context window still even if it is very large, you need to understand that we only have to give relevant chunks.
Right. Relevant chunks in the sense you have your question. Once I ask a question, what do you prefer? Do you prefer relevant chunks or will you prefer all the data? So if I have to give you an example, let's suppose there is a 8 hours long podcast video.

Tirth 5:31
M.

Tarun Jain 5:34
And you have a question which is only related to the last one hour. So do you think the transcript of starting one to six hours is even important? That is just garbage tokens that you are passing it to LLM. So more tokens you pass to LLM, more cost you have to give, right?
So this is the reason why only pass relevant information to the input tokens so that your cost is also less and you're not also hitting the context window limit. So once you have the chunks, still you only have raw data, right? And you can't save raw data into vector database.
So this is where you will convert that into vector representation and we already looked into one of the example of fast embed, right? And once you convert your data into embeddings, you can directly save that in a vector database.
So can you see this arrow mark here?
There are two arrow marks. So when you're saving your data into vector database, you have to pass your data and also as well as embeddings. So basically in this component both the things are happening, the chunks and as well as the.
Uh, converting into embeddings.

Tirth 6:49
OK.

Tarun Jain 6:49
It just force the process, load your data, convert that into chunks. Once you convert that into chunks, save it in a vector database. So when you're creating that object, you have to give your embedding model and as well as the chunk and then you're saving it in a vector DB.

Tirth 7:06
OK.

Tarun Jain 7:06

So now what we'll do is let's understand once you save it in a vector database, how is users only looking like? Just give me one second.
OK, so we'll start with retrieval process. So now let's suppose you have your data, you save that in a vector DB till here you have completed. So whatever you do in this four step, it's happening in the offline process, right? So offline process in the sense.
You already have internal documents of your own data or your customer data and you're saving it in a Vector DB and whatever you do here, no matter how much time it takes, it's happening on your end. So once your data is saved in a Vector DB, this is where you actually build your application right in your application. Now you have a user.
So user will ask a query and let's imagine whatever data you saved, it was related to healthcare. So once you have the query, what was the first step? You convert that into embeddings, right? And once you convert that into embeddings, you will look into vector database.

Tirth 8:21
Things that right?

Tarun Jain 8:26
And then vector database will perform search. So when it performs search it will return the number of KK is nothing but how many tokens you need. I mean how many chunks you need. So every chunk might have 1000 tokens.
It might have 2000 tokens based on what you have defined. So now if you see my K is 1K is 2 just like this, my K value will be 4 right? So if we do the math.

Tirth 8:56
Right.

Tarun Jain 8:59
So you have your data.
And now once user ask a query.
And from this data you're retrieving total K equals to 4.
So what will be the total tokens?
Let's imagine each chunk.
Has 1024 tokens.

So you will have around.
Input tokens.
And this is just user prompt, so whatever you're doing it here right, you'll also have user query.
And you'll also have context.
How many input variables are we passing to LLM?
One and two. Is this clear?

Tirth 10:00
Yeah.

Tarun Jain 10:01
Hello. So basically here if you still see you have where you have context. If you are using memory you will have chat history.
So chat history might again take around 2000 to 3000 tokens.

Tirth 10:21
But.

Tarun Jain 10:21
So roughly, if you just do the overall mathematics right, just with the context and chat history, you have around 8000 tokens.

Tirth 10:29
Mm.

Tarun Jain 10:32
This is just input and then you'll also have output tokens.
So output tokens roughly it will be 1024 which is your max tokens that you define and not every time you will touch 1024, but if you have use cases like report generation that we did in the last time LLM experimentation during that time this will be 4096.
So all these things you have to keep in consideration when you do the cost analysis of any RAG based application, right? So this is only for LLMS, but we'll also have it for other details as well, like embedding model cost, which is very less. Vector DB, you

can host it on your cloud itself. If not, AWS also have a very large.
Marketplace.
OK, so till here what we have understood is user ask a question, you look into Vector DB and you retrieve the relevant chunks and now augmentation is one step here.

Tirth 11:28
I I hi Tarun, sorry I have one question though. If you can go back to the previous slide, we are saying what is what is the best health tip right? And then we are doing vector search to get the relevant chunk.

Tarun Jain 11:39
Huh.
Yeah.

Tirth 11:45
Right, but it can be related. It won't know that it is the relevant chunk. Like how would it know the relevant chunk?

Tarun Jain 11:52
So here it is performing search like cosine similarity. So if you see this particular query is converted into vectors and now in your database you might have this keyword health tip right? So there might be a subsection where in the health tip you had wake up early, you had good diet.

Tirth 11:59
Right.

Tarun Jain 12:11
We'll just extract that particular information via search, which is cosine pairwise calculation distance measurement.

Tirth 12:18
And for that cosine also we use the model right to create that uh uh cosine you know.

Tarun Jain 12:23

No, it's not model, it's algorithm when it comes to cosine similarity. So here there are two things.

One is vector search.

And one more is keyword search.

Tirth 12:38

Hmm.

Tarun Jain 12:39

So in keyword search we have an algorithm called beyond 25.

Or we have PFIDF.

In vector search.

We are using schematic search. We need that the distance measurement is cosine.

Tirth 12:59

OK, but now the if the query changes something to, you know, synonymically like how can I extend my lifespan, which is again asking for a health tip.

Tarun Jain 13:00

Oh.

Tirth 13:11

Then how does the vector search work in that scenario? Because you know I'm I'm just a bit confused about that you're doing the best health tip. It will regarding the health, it will do the search and get the context out of from the vector database.

But uh, if I change the query in a way that you know how to extend my life.

Tarun Jain 13:35

OK, so when you say how to extend the life, if the particular context is not there, whichever is the relevant chunk, right? So if you see here last time also in this projector we picked a particular point.

Tirth 13:49
Mm.

Tarun Jain 13:50
It was diamond or it was tourist attraction. Here there was a different information. So if you see whenever you do cosine similarity, whichever has relevant, what was your question? It was.

Tirth 14:00
Hmm.
How to extend my life?

Tarun Jain 14:07
How to extend my life?

Tirth 14:08
my life. Yeah.

Tarun Jain 14:11
So it will take the schematic meaning of this particular sentence. So whichever is close to this right life and extent, then it will extract information from there.

Tirth 14:20
OK.

Tarun Jain 14:22
Now let's take one more scenario. You will ask what is black hole?

Tirth 14:26
M.

Tarun Jain 14:28
Now nowhere in your PDF you will have a question called what is black hole, right? So technically you will say that it should be empty, but it won't be empty.

Tirth 14:33
Hmm.
Hmm.
OK.

Tarun Jain 14:41
You will have 4 chunks in this.
But how will you measure the issue? The issue is the score. So whenever you perform, let's suppose you ask any question, what is black hole? You're performing search in the search. Even though black hole is not there in my document, it will have 4 chunks.
But what are the parameters you have? You will have source, you will have your page content which is your actual information. Then you have metadata and after you perform vector search you will also have ID and you will have scores.

Tirth 15:15
Oh.

Tarun Jain 15:16
So this course is nothing but schematics course. This one will be less than 50% for sure.

Tirth 15:22
M.

Tarun Jain 15:23
So now what will you do if you want to avoid this on the retriever end itself? Let's suppose you don't want to go to the LLM side and you want to avoid this. Like if you ask anything what is black hole, you want to return empty, so you just have to use similarity.

Tirth 15:36
Oh.

TJ **Tarun Jain** 15:43

Search by threshold.

And then you are you have to define your threshold value. So the threshold value will be 45% or it will be 50%. So if you keep more than 50%, there might be high chances you won't get right results, right? So let's suppose if you keep 60%.

Tirth 15:50

M.

Mm.

TJ **Tarun Jain** 16:06

And you ask the question like how to extend my life. Even though there is some information about the best health tip, when you say extend my life, you might get accuracy score which are very close to 60. It will definitely not be less than 45, but for black hole you will have less than 45.

Tirth 16:08

Oh.

Mm.

Hmm.

Hmm. OK, OK.

TJ **Tarun Jain** 16:24

So we have different search. One is similarity search.

Tirth 16:29

Hmm.

TJ **Tarun Jain** 16:30

Then we have similarity search.

By threshold.

Then we have MMR.

Which is maximum.

Marginal relevance. So here if you look at this search technique, what it will do is let's

suppose you want to travel to India.
And want to visit.
Bangalore.
OK, so this is your user query. Now in the data that you have, you have information on Bangalore and you also have you also have information on Mysore.

Tirth 17:01
It.

Tarun Jain 17:16
Or somewhere with some places which are very close to Bangalore. What similarity search will do is.
It will return 4 chunks. All four chunks will be related to Bangalore.
So what MMR will do is it will apply diverse search.
In diverse search what will happen? First it will take around 10 chunks which are candidates out of them then it will re rank.
And during this time if you have your diverse value, so here there are two values, diverse and relevance. If you pick relevance, it is pure similarity search if you pick diverse.

Tirth 17:59
Mhm.

Tarun Jain 18:00
It will have some information on Bangalore and some information on Mysore. So this way if you are building recommendation kind of systems during that time MMR is very useful because here you have diverse search results.

Tirth 18:14
Mhm.

Tarun Jain 18:15
And for all the things that I mentioned, you will have scores.
So based on scores, you can filter out how close the data that was retrieved.
Is this clear?

Tirth 18:31
Yeah.

Tarun Jain 18:33
So these are just search techniques.
It is applied here at this green color point.

Tirth 18:38
OK.
OK.

Tarun Jain 18:40
So once you add any vector, you apply vector search. Here if you notice I'm not mentioning vector DB, I'm mentioning vector search. So DB only comes into the play until you save that in your data. So once it's saved in the data, when you're applying real time, it's only search.

Tirth 18:57
Mhm.

Tarun Jain 19:00
OK, so now moving to the augmentation part. As I mentioned, most of the time what people do is they skip augmenting their problem. So why do we need to augment the problem? To give more controllability. So this is where if you remember we have a template, the template is.
Always define a system prompt and add a user prompt. In user prompt you have to have all the input variables. So now how will your system prompt look like?
System prompt will have role play.
It will have instruction and then it will have structured response.
Or you can have some 10 motivation.
So this will come inside.
System prompt.
And now in user prompt.
You will have one line instruction.

So what is this one line instruction? It will be answer to the user question.
By referring the given.
Context only if the query is not in the context.
Just say.
Not enough information.
So now why is this important? Let's suppose the user queries what is black hole and as I mentioned earlier.

20:37
M.

Tarun Jain 20:39
When you perform search of what is black hole, you will definitely have 4 chunks. If you're not applying filtering, what will happen is now that particular context is going to user prompt and LLM will check whether for the given query do you have that information in the context or not. If it is not in the context, it will just say not.
Of information. So you'll have one line, one line instruction, then you have query.
Which is user query and then you have context.
Context and if in case you want to include memory, you will have chat history.
And chat history.
Notice all the input variables that I have. Where am I adding it? I'm adding it in a user prompt and we already saw this example when we work with the report generation.
Is this clear?

Tirth 21:35
Mm.

Tarun Jain 21:39
So just make sure whenever you're defining system prompt, you don't include any input variables.

Tirth 21:39
Very much.

Tarun Jain 21:55
I'll take the screenshot.

Tirth 21:58
And just to confirm, I know you already said this, but the context over here in the user prompt is the context of the one that we retrieved from the vector search.

Tarun Jain 22:08
Can you repeat?

Tirth 22:10
So the context that we are passing over here is the one that is, you know, from the vector search.

Tarun Jain 22:15
Correct. Yeah, yeah, I'm coming to that part.

Tirth 22:18
Open.

Tarun Jain 22:20
So if you see it.
You have system prompt, you have user query and now in the generation side, what are you doing now? You have the augmented prompt, then you have relevant chunk. Now you have the large language models. So whatever context you're getting from generation side, it will be appended to the user prompt.

Tirth 22:42
Mhm.

Tarun Jain 22:43
So this context, whatever you have here, it is coming from vector DB.

Tirth 22:47
M.

Tarun Jain 23:01
So the task of user it is just to give this user query.
This is the only user query you have. This context it is extracted from Vector DB.
So your application will be you ask a query. So if you notice this particular diagram, what are you passing in generator in the augmentation?

Tirth 23:30
System correct.

Tarun Jain 23:30
So if you see you have system prompt, you have user query, but what is there in user query?
Just one input variable. You still don't have the context. So when you build the generation pipeline, in the generation pipeline in Langchain, you call it chain. You might have already seen Langchain expression language, the pipe symbol. So when you create a chain mechanism.

Tirth 23:41
Um.
Hmm.

Tarun Jain 23:55
You will pass user query and then in the chain you will also define for the given user query, get the relevant chunk from the retriever 1st and then give it to the LLM model. So now LLM will have system prompt, user prompt and whatever this thing is there.
Context it is appended in the user prompt and not in the system prompt and once it is passed to the LLM model you have your final response.
So how do you append this relevant chunk? We'll see that in the code.

Tirth 24:27
But.

Tarun Jain 24:29
But till here, is it clear?

Tirth 24:31
Yeah.

Tarun Jain 24:32
Just to summarize, there are two steps here. The first thing is document processing and 2nd is answer generation. When you're performing document processing, just ensure it happens in the offline mode. You upload your data.
And chunk it. And once you chunk it, you create the embeddings for it and then save it in a Vector DB. Once your data is saved in Vector DB then you can build your application. This application can be your web app, it can be a slack bot. Let's suppose.
You want to build an internal document chatbot for the documents that you have and you want to deploy that in a Slack chatbot. What you can do is save all your data inside Vector DB, then build a Slackbot interface where user will take the query.
You convert that into embeddings, check from Vector DB. It will retrieve the relevant chunks. Now LLM will get the two response. One is chunk, one more is query and LLM will generate the final response. This response is again showed to the UI where it can be our web app or it can be our slack chatbot.

Tirth 25:28
Oh.
Mhm.

Tarun Jain 25:42
And this is very important to notice. I'll also insert that image here.
So system prompt will have role play instruction and structured response or motivation. Input variables will be defined inside user prompt only and this is how that one line instruction looks like. So answer to the user question by referring to the

given context only. So context is the only source of information.
That your app has. So even if you ask who is Virat Kohli, if you get any context here, your LLM will ensure if the context matches with the query or not. If it doesn't match us, it will just say I don't know or not enough information. You can decide what fallback.
Prompt or fall back message you want to show to the user for. In my case I usually keep not enough information provided.
So it's better. Sometimes LLM doesn't give the response, but you have to ensure it doesn't give false responses.
Is this clear?

Tirth 26:50
Yes, it is.

TJ **Tarun Jain** 26:51
OK, so I'll share the slides. Uh, these are the slides that we are using it from starting.
And in order to begin with the code, I've not written any code yet, So what we'll do is we'll write it live itself. We can create a new collab notebook.
And I will also attach this particular slides in the GitHub URL that we have.

Tirth 27:30
OK.

TJ **Tarun Jain** 27:36
So for offline document processing, once we complete components, right? Once we complete the components.
We will start with Raj.
So the document processing we will do it on collab.
And the answer generation part, whatever you have this side, we will do it on VS code.
So we'll start with document loading. We'll test with PDF and there are other components as well like PPT is there, Docs is there. We will try to experiment with that and these two we already covered. One is the large language model and chat models and as well as the prompt template. We completed this during the LLM experiment.

Part and then we have structured output responses. Does anyone know what library we'll use here?
For structured output response.
We'll be using pirantic.

Tirth 28:56
Yeah.

Tarun Jain 28:58
Pyrantic based model where we will define what keys we need and the description for that keys. Now what LLM will do, it will look at that key, it will look at the description that we provided and then it has to generate the value for it. And now if you have the key value pair, you have your JSON response.
So this is where we'll be using pyrantic based model.
Then tool calling we will try to ask any question. We'll ask real time query and based on the real time query it has to look into web, use search and get the result and embedding creation also we already covered. We used fast embed vector database will take some more time. We'll try to understand different operations.
That we can do from vector database. So this will take time. We'll take one or two sessions for vector databases as well because this is very important. Not every time you will use the default embeddings or default object from line chain. You will have to use your own client, you will have to.
Use some configuration. What are those configuration? We will discuss that when we dive deep into vector database like advanced topic. But for before we jump into rack we have to cover these concepts.
OK, uh, so have you created a new collab notebook?

Tirth 30:47
Yeah.

Tarun Jain 30:47
OK, so so far I don't think we installed any libraries on collab apart from pip install fast embed.
As most of the libraries that we had so far, uh we were just utilizing the prebuilt libraries that we had in Collab. So now what we can do is we can use pip install.

Launching.
Then Langchen community. So from Langchen community we will be using the document loaders.
Then if you remember, everyone of us used LM's like.
From lunch in Google Geni. So this is mainly for chat models.
OK.
And for prompt template we just need the launching code.
And Pyrantic is already there, so we don't have to install any specific library for structured output response. So one new thing what we will do today is last time if you noticed you guys had to wait in order to get the final response, right?
Wait until you get the final response. So today we look at 2 approaches. First is invoke. So what does invoke do?

Tirth 32:07
It will do the same thing. It will wait till the end for the response.

Tarun Jain 32:11
Generate response once it is done.

Tirth 32:16
Yeah.

Tarun Jain 32:19
Then we also have something called a stream, which is 1 token at a time. This is like ChatGPT. So you ask any question, let's suppose.
Write an essay.
On live.
So here if you see it's like one word at a time. This is mainly streaming, so we'll look at how to do streaming. So what invoke does invoke will only print response after it sees this end. So once you see invoke is like you have to wait till the end. Stream is like you can read it while it is generating 1 token at a time.
So that can be done using LLM.
And apart from that, what is missing?
Pip install.
Huh. I wanted to show something else as well.

So this is the architecture diagram of Uber. I just wanted to relate the concept of document processing and why it is very important. So you're able to see these slides.

Ronak Makwana 33:36
Yeah.

Tarun Jain 33:36
Hello OK so this is the architecture diagram of Uber. So if you notice here once user ask for query it will look into the slack chatbot and then you have the answer generation part and if you notice the center part this is only for answer generation. And if you look at the bottom part, it is for document processing. And what is happening in document processing, this is happening in the offline mode. So first what they're trying to do is they have their knowledge documents, which is the internal documents, they convert that into Google Drive.
And once they convert into Google Drive, they have a proper HTML formatting. I mentioned this last time also. Whenever you're passing any information to LLM model, XML, Markdown and HTML formatting works very well. Even these people are using Markdown. So once you extract it in HTML format, they convert it into text. Which is in markdown. And here when you look at this approach, why didn't they use any basic PDF, right? There are so many loaders. If you look at this particular problem statement that they had, they experimented with five PDF plumber.
Pi MU PDF, which we will do today and they also experimented with llama parse. So when they experimented with all this PDF loaders, they were not able to extract all the information well. Now why is this happening when you have your data?
Your data will have tables and you need to extract proper information from the tables. So this is where having a proper extraction tool is very important and this is where what we usually do is I have done some kind of research. There is this particular paper called.
Comparative analysis for PDF parsing. So here if you notice you have PDF miner, then you have PDF plumber, then you have Pi MU PDF which we will use now and this four things is something that we will use Pi PDF plumber.
5 MU PDF, 5 PDF and five PDF form and based on this there are some of the data set which they have used and they came up with this particular scores. So now if you see in financial domain, law domain.
Patent and Scientific This particular loader Py PDFM have higher scores compared to

the other loaders.
So let's start with the installation.
Install py PDF.
So this is mainly for loading the data. You have pipe PDF.
And what else do we have? We have Py MU PDF, so I'll just search for Py MU PDF line chain.
Buy a new PDF.
And then you also have 5.
PDF um.
And you have to install this. You understood what we're trying to do.
So basically when you extract information from PDF, you just have to remember this code.

Tirth 36:51
Yeah.

Tarun Jain 36:57
More than quality of model building, the data that you extract is very important. The data in the sense the quality of the data. And in my experimentation, let's suppose you have a table. Inside this table you have a value called 25 degrees Celsius.
So I don't know how to add. So let's imagine this is degree. I'll just write degree. OK, so when you extract information from table, sometimes this 250 degree is sorry, 25 degree is converted into 250.
And now if you ask any question where you want to know what was the degree Celsius, I'm just telling an example and during that time you are getting the response like 250, which is completely wrong. So it is very important to understand which is the proper extraction tool we are supposed to use.
And based on the results that we have seen from the research, these two are pretty reliable. One is Py PDFM and one more is unstructured I/O. So if you see unstructured I/O is also very close to Py PDFM and this is mainly also used for OCR. So OCR in the sense if you have data and if the data is not textual, it is images right? Mostly we have scanned PDFS and if you want to extract information from scanned PDF, unstructured IO is much better than any other loaders.
Right. So we'll also use unstructured loader, but the only thing is unstructured loader is pretty heavy model. So for that we also need OCR. We will cover that when we use

multimodal RAM, we will cover.
Unstructured IO when we discuss.
Multimodal drug.
The procedure is same. You first have to load the loader and then extract the data. It's just three lines of code, nothing much. But the only thing is this requires two additional library to use this particular what you call framework. So this is the reason why we'll not use it today. We will cover it once we have some basic under.
Standing in place. So now what you can do is you can just run this and you can cross check the spelling. It is Py PDF, Py MU PDF, then Py PDFM 2.
And when you run this right launch in Google Gen. AI, it will tell you to restart runtime. Don't do it until you see green tick mark.
Or does anyone have any question?

Tirth 39:43
Can you just copy paste the names in the chat by PDF by move PDF which one?

Tarun Jain 39:47
Uh.

Tirth 39:51
Yes.

Tarun Jain 39:56
I'll also share these slides. It's very useful.

Tirth 40:00
Thank you.

Tarun Jain 40:11
And this is the blog article.

Tirth 40:22
Hmm.
Mm.

Tarun Jain 40:24
But the only thing is they added this additional thing. This is hybrid search. If you see you combine vector search with the keyword search which is happening on the retrieval process, even we will cover this part and this is very straightforward.

Tirth 40:30
Beam 20, yeah.

Tarun Jain 40:38
And if you notice they have also added additional custom metadata. Why they added their custom metadata? This is what I will explain during Vector DB. So what we will do is we will use the same components what line chain provides, right?
But vector database is 1 important topic where you want to improvise. There are multiple customization which you can't do with any frameworks. Why? Because these frameworks are just wrapper. If you want to add your own support, you want to know how to use each and every single individual component so.
In all the RAC pipeline, the most important part is your vector database. So this is the reason why we will have a separate session on vector database and understand what are the different optimization techniques that we can do.
So what is the next step after you install the library?
What do we do next after you install library? Does anyone recall?

Tirth 41:45
We'll first get the context and so we'll first start converting data.

Tarun Jain 41:48
We keep on runtime, then restart session.

Tirth 41:51
Oh, OK. Thank you.

Tarun Jain 41:52
Why? Because there is a dependency issue of launch and Google Gen. AI and Google Gen. AI, which is already there in the collab.

Tilia, is it clear? OK, so now what you guys can do is if you have any PDF with you, you can just upload that. It can be any PDF.

Tirth 42:02
Yes.
I have one for career leveling guide. It's a 300 page PDF which people might want to use OK.

Tarun Jain 42:17
Ha, that's fine. We are not doing rag as of now, we are just loading the data.

Tirth 42:24
I'm adding that in our chat so everybody can use it if they want to.
You guys can create a chat bot around career leveling guide in future, you know, and share it with me once it is done.

Tarun Jain 42:44
So if you're using any PDFs, right, let's just make sure we'll use the same PDF for the rag as well. So now what I'm trying to do is I want to build a policy agent. Sorry, not agent policy chat bot.
Ner So this is the document that I'll be using National Education Policy 2020 by Government of India. It has table. I actually want a table so that I can test how the extraction works.
So I'll also share this if anyone is in.

Tirth 43:15
I I have a table of content in career leveling guide as well.

Tarun Jain 43:19
Cool. We can use any PDF, but let's just make sure whatever chatbot you build, it is something useful where you can relate or you can pitch into any demos.

Tirth 43:22
OK.
Big.

Tarun Jain 43:31

OK, and I'll also expect anyone to build a chatbot either for Microsoft or it can be for Slackbot so that most of the things are done in a very simple way. The only thing what you have to focus on is the interface.

The rag backbone is same. User will ask a question, look into search, get the context, get the response. So where user will ask a query and where user will see the response that interface someone has to build and this interface can be.

Tirth 43:54

M.

Tarun Jain 44:03

Chat bot or it can be a web application.

OK, so is your file uploaded here?

Tirth 44:11

Yes.

Tarun Jain 44:11

So now you just write path.

Copy this.

And first thing is from lunching.

Community dot you have document loaders.

I will import by.

I will import py PDF loader. I will also use py MU PDF and I will also use py PDF.

So these are the three things I'll be using.

So this is very basic. OK, don't ever use this pipe PDF and this is fine and this is better compared to whatever we saw here.

Have you just noted this particular line?

Let me know if it's done.

So it's just three-step process. So the first step is we import a loader. The second step is we define a loader. The third step is we extract the raw data. So we just now completed 2 steps.

And now what I will do is I have data one equals to loader one dot load. There is a

function called load. I will repeat this for the other as well, but let's just make sure we run it in different cell because you have.
Uh, very large PDF and probably everything might take some time.
Loader 2 dot load. It was like hardly two or three seconds. I I don't think it should take more than three seconds.

Tirth 46:33
M.
We are doing.

Tarun Jain 46:49
Pi PDF. If you notice this thing Pi PDF loader, it took 14 seconds, whereas Pi MU took 0 seconds, then Pi PDFM took one second.
Let me know if you also got the green tick.
For future projects.
Use unstructured IU or llama parts.
If OCR based, if not use.
Fine.
PDF M.
Or.

Tirth 47:58
I got a warning for loader 3.

Tarun Jain 48:01
That is some deprecated issue, but it should be fine.
So I've added one note here. If you look at this note, if your data as OCR right or AV tables, tables like you have 5-6 tables and most of the research papers you have figure below the figure you have diagrams.

Tirth 48:22
Um.

Tarun Jain 48:22
During that time use unstructured IO or use llama parse if it is OCR based. Now why

did I add R here? Unstructured IO is like open source. Llama parse is API based. There is certain fees credits that you have to get and then use llama parse.
And if you want to use completely open source and unstructured IO, if you want quality and if you need fastness then llama parse. And if you don't have OCR based and if your data is only texture data then you can use spy PDFM or unstructure IO. Anything between these two works.
So now what you can do is there are two things that a data will extract.
One is the raw data.
Which is our page content?
And the second one is.
Metadata.
And metadata is very important as well. Now I'll tell you one use of metadata, let's suppose.
You have 10 PDFs, 10 I'll say 10 policy PDFs. Now if I ask any question from which policy document was the response generated, first thing I want to know the source. Second, I want to know from which page number it generated. So you need to know the page number and as well as the source. That information is there in the metadata.
So now if I print data.
One dot first you can check the data type of this. It should be list.

Ronak Makwana 50:08
Event.

TJ **Tarun Jain** 50:11
So if I do length of data one.
If you see it is 66, that means for every single page my data has around 66 pages, so it has just created one chunk for every single page. So what I'll do is I will do 0 index and then here you can use two functions.
One is page content.
Which is your raw information. If you see in my first page I had National Education Policy 2020, Ministry of Human Resource Development, Government of India. If I use 5th index probably my I might have different thing now if I use metadata.
You will see you have source.
You have page number. Page number is very important and there are other

important fields, but I don't think producer is needed, creator is needed, creation date is needed or even author. These four things are even modified it this file.
Will not make sense. So when we use RAG, we will just remove this information from metadata. We will only keep source, total pages and page number.
You can do the same thing for data too. And if I do metadata, if you see it for Pi MU PDF you have more additional fields, you have format, you have some keywords and then you also have subject.
And you also have title. Even for pipe PDFM you will have title and all.
You have title subject, but most of the time it will be empty.
For 0th index.

Ronak Makwana 51:52
OK.

Tarun Jain 51:58
And my first index should be a table of content.
So if I do print, it will print in the straight line. You can also test it with data one and cross check how well the information was extracted.
Chin Chinmuji Bharat Song.

Tirth 52:29
For outside the TOC starts from.

Tarun Jain 52:32
Oh, what happened?

Tirth 52:33
3.
But we have multiple pages for TOC term, the table of content, so it will still work as this one.

Tarun Jain 52:44
Multiple. If you're having multiple pages, usually what we do is we have to extract it based on HTML, right? So that HTML thing is supported in unstructured IO.

Tirth 52:55
Um.

Tarun Jain 52:59
Not HTML, but basically it is markdown, not HTML markdown.

Tirth 53:03
OK.

Tarun Jain 53:06
So are you able to extract the information? So this is very important and I hope this is clear what is page content and what is metadata. So when you save your data into Vector DB, you will mainly use page content to vectorize.

Tirth 53:11
Yes.

Tarun Jain 53:25
So there are two concepts that we need to understand. I will explain the two concepts here. I will just write a note so that I won't miss it.

Tirth 53:25
Yeah.

Tarun Jain 53:34
Vectorization of page content and payload of metadata. I'll talk about when I discuss vector database.

Tirth 53:50
Mhm.

Tarun Jain 53:52
Is it clear why we need metadata and what is page content? Metadata is like if you have multiple documents and if you want to build applications where you want to

cite, hey this is the response and I got the response from this page number and this source.
So during that time this is useful.

Tirth 54:07
Hmm.

Tarun Jain 54:13
Which is nothing but your metadata. And what is the data type of metadata?

Tirth 54:19
That is.

Tarun Jain 54:21
It is in dictionary.
Is this clear?

Tirth 54:27
Yes.

Tarun Jain 54:28
So now if we upload documents right, in most of the cases it is very rare we will get page number for documents. So we can also test it if anyone has docs file. So for docs file what we can do is we have different loaders.
You just have to use from land chain dot.
It's community dot document loaders.
Import if you just type unstructured UN. If you see you have file loader, then you have HTML loader, then you have PDF loader, PowerPoint loader, then Word document loader. If you just add DOC.
There has to be.

Tirth 55:12
When sorry, when you do image loader from this, does it do OCR by itself or no?

Tarun Jain 55:18
So unstructured IO it will do OCR.

Tirth 55:23
OK.

Tarun Jain 55:24
So if you see the image loaded is from unchecked IO so it uses pytestra.

Tirth 55:27
OK.

Tarun Jain 55:32
And if you want to use pytestract, what is the simple command? You might think it is pip install pytestract, right?

Tirth 55:39
M.

Tarun Jain 55:40
But basically this is a package. It is like you have to do pseudo install and all.

Tirth 55:40
Yeah.
Yeah, that's it.

Tarun Jain 55:45
So this is the reason why I don't wanted to cover that today. So it takes at least two libraries for OCR. In Linux command we have something called as poputils. I'm not sure if it is the same thing.

Tirth 55:48
OK, OK.

Tarun Jain 56:00
Pop to this.
There is some command where you have to install some utils file, but I do have that code available of unstructured I/O. So if you have any data right, let's suppose you have your data saved in Airbyte. So if you also have Airbyte and what else do we have?

Tirth 56:19
OK.

Tarun Jain 56:30
We usually use this thing Azure BLOB. So what happens is in Azure you can create a BLOB storage and then you can define this Azure BLOB storage and then there are certain credentials. So the credentials is the container name.
What is the container name? You have to give that name and then you will also have certain endpoint URL. So you have to define that endpoint URL and get the information. So for that you can use Azure BLOB and for some of the enterprises you also have Microsoft SharePoint.
You have SharePoint loaded.
So now the question is how do you get to know what credentials to use? So just copy this line, come to documentation. If you see Microsoft SharePoint here they will tell how to get the actual credentials. Here you have tenant name, collection ID, substitute ID.
So the document loaders documentation, what Langen has written, it's very specific for all the credentials, whatever you need, you can check out this page.

Tirth 57:34
Mm.

Tarun Jain 57:36
Never use CSV loader. They do have CSV loader, but don't use it. What should you use instead of CSV loader? What library?

Tirth 57:47
Uh, the pandas.

Tarun Jain 57:48
It should be pandas. So if you see you also have website, let's use website.
I'll come back to the page so document loaders. I will use web-based loader.
Do what Atyantik website allow scraping?

Tirth 58:09
Yeah, it does. You want everyone to know about ourselves. 80 by AN. You missed. Yeah.

Tarun Jain 58:20
OK, so let me copy this URL.
So let's suppose you want to build a chat bot for your own website. First thing is you use web base loader. What is the next line?

Tirth 58:35
Loader equal to web-based loader and then you give pass HTTP yes.

Tarun Jain 58:40
web-based loader and here if you see you have web path instead of file path.

Tirth 58:44
But yeah.

Tarun Jain 58:47
You'll just pass the website URL.
And then I'm defining data. If you notice earlier I've used loader 1, loader 2, loader three, then data one, data two, data 3. So I'm not overriding any information. Loader is a new variable only data equals to loader dot load.

Tirth 59:02
OK.

Tarun Jain 59:10
Now data length of data.
It is one data dot page content.

Tirth 59:17
Hmm.
Data 0.

Tarun Jain 59:28
So you have the information.
So let's suppose you ask any question like where is Atyantik located? What? How do I contact Atyantik? So during that time what will happen? This particular chunk will be there, it will go to your metadata and then LLM will look and get this particular context. What is the contact number and what is the e-mail?

Tirth 59:52
Wait.

Tarun Jain 1:00:03
And if I use metadata.
You have source, you have title, software development company in Varadra, Gujarat, Atyantik, then you have description and then you have language.
So this what you can do is you can repeat it for multiple pages. Now I'll click on about us.

Tirth 1:00:18
Mm.
Yeah, it's loading.
Did today.

Tarun Jain 1:00:36
I will copy this.
LRDURLS.

OK, this page doesn't have anything like if in case I pass this, there is not much of the information because it will not extract images.

1:01:11
Right, right.

Tarun Jain 1:01:13
Yeah, this is image, so I'll pick.
Where do you have more text like which particular?

Tirth 1:01:20
Software development or SAS development? Yeah, in development you can go to SAS development.

Tarun Jain 1:01:26
So now what you can do is you can get all the URLs from services which you have in development and build a chatbot for that.

Tirth 1:01:35
OK.

Tarun Jain 1:01:52
Yeah, it's not that much.
But yeah, this this should be enough for time then.
So now what will I do here? Instead of passing one path, I'll pass this URLs.
So now what should be the length of data?

Tirth 1:02:17
644.

Tarun Jain 1:02:22
What should be the length?

Tirth 1:02:24
4.

Tarun Jain 1:02:26
It will be 4.
Data one it is about us.

Tirth 1:02:31
Hmm.

Tarun Jain 1:02:34
Then data two that is related to the services, then data three again it's related to services and if you see these two are getting changed every time.

Tirth 1:02:46
Hmm.

Tarun Jain 1:02:47
Now I'll tell you one hint what metadata can also be helpful as. So let's suppose you have some information.
Where you have information in two different web pages, but you want to filter and want to extract information from a specific page number. So during that time what you can tell is you can filter out whatever you want to extract from vector database by defining your title as as development service.
So you understood what I meant to say. So you have software development, you have SAS development. Now user ask a question. You have 4 context which you're extracting. Let's suppose you have K you're extracting 4 and this two are generated from.
Software development and two are generated from SAS. But what you want is you know that this response will belong to SAS itself and you want to apply certain filtering. So during this time what you can do is you can use your payload which is saved in Vector DB.
And define your filter where title you can match with SAS.

Ronak Makwana 1:03:54
Mhm.

Tarun Jain 1:03:55
So now what K will be is K will be just two which is coming from SAS and you're not confusing the LLM. So full time technique is very important to reduce allucinate and also to improve the retrieval performance.
OK, Killier, is it clear?

Tirth 1:04:16
Yeah.

Tarun Jain 1:04:17
So now if you notice in the PDF.
If I do the length of data 3.
It should technically be one, but it is 66. The chunking is already happening over there, but here the chunking is not happening in the website. So now what you can do is you can define from land chain.
Can you see this lanch and text splitters?

Tirth 1:04:46
Yes.

Tarun Jain 1:04:46
Click on that and then just use recursive.
Character text splitter.
And now you have to define your splitter.
It was so.
Here there are two parameters you have to define. One is the chunk size.
I'll keep it thousand and then you have chunk overlap.

Tirth 1:05:16
Hmm.

Tarun Jain 1:05:19
So let me tell you what chunk all up is. Let's suppose you have 4 documents, which is your website.

And I'm just assuming this four will create around 25 chunks.
I'm just assuming it. So now when you create these 25 chunks, let's suppose this is your chunk one.
This is your chunk too.
This is your chunk 3. Now if you need any information from chunk one to pass in the chunk 2, let's suppose you have one statement at the end.
So you have your information and in the last time it says this is.
The end. So this is just a placeholder. If you want this line, this is the end to be repeated in the chunk 2 beginning. That is when you define a chunk overlap. Chunk overlap is nothing but how many number of tokens do you want to pass in the consecutive chunks?
Right. If you want to happen, then you can define some code up to B.
100. So now what will happen in chunk colab to be 100? The last 100 tokens of chunk one are the 1st 100 tokens of chunk 2 is what chunk colab is. If you don't need any overlap, you can also define it as zero. In most of the ideal scenarios it is like this.

Hardip Patel 1:06:46
Mhm.

Tarun Jain 1:06:55
You define 2048 and chunk or lap will be 0.
If not, if your data is very small, you can keep it as 1024 and your chunk or lap is 200.
But again, there is no proper mathematics behind it. Most of the people, they just play around with these numbers and I usually keep it as 2048 and I keep chunk all up to be 0 because I don't want redundancy in the chunks that I have. So I keep it 0 and this is 2048.
So now what you have to do is you create chunks.
Splitter dot you have split documents. Now why do I use documents if I print data and zeroth index?
What is this object?

Tirth 1:07:49
Up to me.

Tarun Jain 1:07:50

It is document. Inside document have metadata and they have title. Is this clear?
Sorry, you have metadata and you have page content.

Tirth 1:07:56
Yeah.
But.

Tarun Jain 1:08:00
So there are only two attributes inside document. So now what do I need to do? I want to create chunks by splitting via documents and what is the variable data?
And now if I print length of chunks.
It is 45.
Length of data is.

Tirth 1:08:22
4.

Tarun Jain 1:08:22
4.
Is this clear?

Tirth 1:08:25
Yes.

Tarun Jain 1:08:27
So far what did we cover? We covered document loading and splitting in most of the PDFs. It's already configured by Lancet folks itself where the chunking is happening. But if you don't need the chunking and if you want to get more in detail then you can use separate splitters and here if your data.
Is in markdown. What you can do is there is also markdown splitter. You have markdown text splitter. You can use markdown text splitter where based on the hashtag which is which is your eddings you will split your data.
The syntax is same. You first have to define that loader, check what kind of data it accept. If it is web path, you can define web path. If you're using Azure BLOB or Confluence, they need credentials. So when you define your loaders, either you

define path or you define credentials.
Once the loader is defined, the only thing is you have to extract the raw content by using loader dot load. If you are doing this on a website then probably you will have async function as well. If you see you have a load, a load is nothing but it's happening asynchronously.
Is this clear?

Tirth 1:09:42
OK.

Tarun Jain 1:09:43
Text splitter is also same. You first have to define chunk size and chunk overlap and once this is done you just have to define split documents, split documents because your data is in a document object, pass your data and then just.
Take the length of the chunks.

Tirth 1:09:59
Mm.

Tarun Jain 1:10:02
Chunks of 0 will is also a document so you can get page page content.
And also metadata.
Cool.

Tirth 1:10:18
Mm.

Tarun Jain 1:10:20
OK, so let's proceed with the language model. Here we have to experiment with two things.

Hardip Patel 1:10:31
I have one question.

Tarun Jain 1:10:33

Oh yeah.

Hello.

Tirth 1:10:40

He'll he'll be back. He's just.

Tarun Jain 1:10:44

What happened?

Tirth 1:10:44

Getting to place. He'll be back. Uh, they're all in the same room now, so it is echoing. So he'll ask the question in a minute. Yes, I did. Go ahead.

Tarun Jain 1:10:52

OK, OK.

Tirth 1:10:59

You're not audible, Mitesh.

Tarun Jain 1:11:10

Oh, now it makes sense why many people not answering. Now you're not interactive because of the call.

Tirth 1:11:11

I will be.

Hardip Patel 1:11:15

Yes.

Hello. Hello.

So what I want to ask is just a second.

What if I want? So we are chunking and I want to add extra data to each chunk, let's say chunk one, chunk 2, chunk 3, so that I have better understanding because this metadata will be same for this page, right?

Tarun Jain 1:11:47
You can add the custom metadata as well, so if you see it.

Hardip Patel 1:11:49
Uh.

Tarun Jain 1:11:53
25th.

Hardip Patel 1:11:55
Hello.

Tarun Jain 1:11:56
Yeah, yeah. Can you see this?

Hardip Patel 1:11:56
Yeah, anything.
Yes.

Tarun Jain 1:12:01
So basically this is all your metadata includes, right? You have source and you have page number.
Hello.

Tirth 1:12:08
Yes, we can hear you. Yes, we are hearing you.

Tarun Jain 1:12:09
OK, so let's suppose you want to create your own custom metadata. You add source, you add page number, and then you are also creating document summaries. So this document summary is your own logic that you are creating. And then what you can do, you can just append it in the particular.
Metadata. So here what we are trying to do is document summary is same across all the chunks. Then you have chunk ID, chunk frequent class questions and chunk

keywords which is again generated by LLM. If you see these three components are generated by LLM.
And this will be unique for each chunk and you can modify chunks metadata as per your choice. It is configurable. So what is the format of this? It's a dictionary, so dictionary is mutable.
So let's suppose unkov 0.

Hardip Patel 1:12:59
OK.

Tarun Jain 1:13:03
I know this is the home page, right?

Tirth 1:13:08
Hmm.

Tarun Jain 1:13:09
I'll just tell some 0 metadata.
And I want to add a new field called.
Is homepage.
I'll just keep it true.
So now if I print chunks of one dot metadata.
If you see there is no E's homepage, but if I do chunk zero of metadata, I have E's homepage to be true. Now you just have to run a loop.

Hardip Patel 1:13:36
Mm.

Tarun Jain 1:13:51
If some of source double equals to.
Ha, this is correct. If you notice, can you see this logic?

Tirth 1:14:02
This.
Yeah, just double equal to.

Tarun Jain 1:14:05
Huh. So what I'm trying to do? I will run for every single chunks. If the source belongs to atyantic.com, that means I need to add a field called ease homepage to be true. If not, I have a field called false.
Order.

Tirth 1:14:25
Chunk uh chunk not metadata.

Tarun Jain 1:14:29
Sunk dot source.

Tirth 1:14:33
OK.

Tarun Jain 1:14:34
OK, wait, wait, this is chunk dot metadata.
So now if I print chunk of 10th index.
dot metadata.
This is not from home page, it is from about us page. Is this clear? So metadata you can even these folks have done it.

Hardip Patel 1:14:58
Yes.

Tarun Jain 1:15:04
They added their own custom metadata and that custom metadata is nothing but document summary, chunk ID, chunk frequently asked question and chunk keywords. This is again going in. Metadata is always going in payload which can be used for filtering and keywords is very important for filtering.
Which I'll cover when we have a database. I've added the note.
Still here, is it clear how to load the data, chunk the data and as well as modify the metadatas. Metadata can be modified because it is in dictionary and dictionary is mutable.

Tirth 1:15:33

Yeah, yeah.

Hardip Patel 1:15:34

Yes.

Tarun Jain 1:15:46

OK, so here, can anyone tell me what will be the first line?

We already covered large language model.

Import.

Tirth 1:15:59

We have to 1st initialize the LLM model, so we have to import the LLM model that we are looking for.

Tarun Jain 1:16:05

OK, from line chain.

Tirth 1:16:06

10.

Import Google Chat, you know, chat model something. It was chat. It was something with chat. Chat generator, OK.

Tarun Jain 1:16:14

Google.

OK, next step.

Tirth 1:16:20

Next step, we would define the system prompt and that syntax of it. So we need the system prompt, we need the user prompt.

Tarun Jain 1:16:29

OK, that will add in prompt template. Here we can just have user query.

Tirth 1:16:33
OK.

Tarun Jain 1:16:34
Because from template also valid as a separate sub ready.

Tirth 1:16:44
Mhm.

Tarun Jain 1:16:48
OK, next.

Tirth 1:16:56
Initialize the model, initialize the.

Tarun Jain 1:16:58
We got very important step.

Tirth 1:17:01
Unless you like it.

Tarun Jain 1:17:05
OK then.

Tirth 1:17:05
We have to pass the key and.

Tarun Jain 1:17:24
I'll keep it unit.
So where should I define the key? It should be like import OS.

Tirth 1:17:44
OS dot environment.

TJ **Tarun Jain** 1:17:44

Then.

Do you have this?

Saved to those who have used the what do you call Samba Nova last time. What you can do is you can use Samba Nova. If not, we can also use Google's and the TVI today.

Tirth 1:17:49

Yeah.

TJ **Tarun Jain** 1:18:02

So to those people who created Samba Noah last time, what you guys can do is you can open aistudio.google.com then click on Get API keys.

Here you will see a button create API key. You have to click on this create API key.

And then for you you can see create API key in the existing project. You just have to select that. In my case it's already created so I'll just click on this copy. Just notice first two keywords is AI.

Then come back here, click on this.

Secrets. Write Google API key, then paste your API key.

Do you guys followed yeahstudio.google.com get the API key? Here you have to click on Create API key.

And then create API key in existing project. Once you do that in the bottom you will notice your API key.

Let me know till here if it is done.

Tirth 1:19:40

A little time till here.

TJ **Tarun Jain** 1:19:44

If you want to use Sambanova, you can also use Sambanova.

So after structured response, I will add one title called LCEL. Does anyone remember what is full form of LCEL?

Tirth 1:20:41

I remember it was for piping.

Tarun Jain 1:20:44

We use pipe symbol that is correct.

Tirth 1:20:46

For Langin, for Langin, something with Langin and piping.

Tarun Jain 1:20:50

Lang first word is correct, second term is also correct.

Tirth 1:20:53

Yeah, I don't know. I don't know yet. Endline. Endline. No, no.

Tarun Jain 1:20:56

Else is.

No.

Tirth 1:21:02

And the last one is language. No, no. OK, then we are only waiting for E.

Tarun Jain 1:21:05

Language.

Russian language.

Tirth 1:21:10

OK, length and expression language.

Tarun Jain 1:21:14

So this is short form for creating chain. So earlier what we used to have is we had LLM chain. So what we used to do was we define from.

Tirth 1:21:17
OK.

Tarun Jain 1:21:25
Blank chain import.
We had this LLM chain. Then inside LLM chain you have to define your prompt LLM.
So now instead of doing these two lines, you just have to do prompt.
Pipe LLL. That's it.
So this is the reason why many people in starting never wanted to use line chain because it was too much of heavyweight. They had too much of dependency, too much of wrappers, which was unnecessary. After they released LCL and then Landgraf, much of the details started coming in place like.
They removed most of the dependencies issue and it started becoming standalone framework agnostic. If not before it was disaster like all this LLM chain. So if you see there are multiple chains you have react chain.
Did I tell React keyboard before?

Tirth 1:22:27
No.

Tarun Jain 1:22:27
OK, I'll skip this then. There was also something called a sequential chain.

Hardip Patel 1:22:28
No.

Tarun Jain 1:22:34
It's in the second line only.
So if you see self as with searching, there was so much of chain and none of them was required. Now we just prompt then LLM and if you need structured response you can have structured response.
So even this is important, I've just added that subtitle. So is this done?

Tirth 1:23:01
Yeah.

Tarun Jain 1:23:01
Can you cross check if you have completed this?

Tirth 1:23:04
Yes.

Tarun Jain 1:23:06
So first we will try with invoke. So what is the command for invoke? Just use LLM dot invoke.

Hardip Patel 1:23:06
Yes, yes.

Tarun Jain 1:23:14
Then query and then print response dot.
Content.

Tirth 1:23:22
But we still have to give prompt now.

Tarun Jain 1:23:26
Uh, this is the prompt, no?
So we'll define from template.
So this is the prompt only, but the only thing is we don't have system prompt here.

Tirth 1:23:38
OK, OK.
So we're just asking the Gemini 2.5 Pro with the existing data that it has.

Tarun Jain 1:23:46
Ha, this is 0 show prompting.

Tirth 1:23:49
OK, OK.

Tarun Jain 1:23:53
OK, not bad.

Tirth 1:23:56
Yeah.

Tarun Jain 1:23:56
I don't know if this is right information or not. You can just.

Tirth 1:23:58
That is, that is right information. Uh, just the location is wrong, but otherwise uh, the rest of the details is right.

Tarun Jain 1:24:08
Full.
So now what we can do is as you can see it took 19 seconds and it generated the lengthy response. I had to wait until it generated India. So now what you can do is.

Tirth 1:24:14
Mm.
Um.

Tarun Jain 1:24:24
Create a new code cell, just write for some OK this type thing. So whatever it generated it is correct, but I'll still write it. So what we need to do is instead of invoke we have to do stream.

Tirth 1:24:34
Hmm.

Tarun Jain 1:24:40

Then query. Let's suppose I run this query outside.
I'll then response equals to LLM dot stream query. It's the same thing what I've done here with invoke.
So now if you see it generated in 0 seconds, but what is this response? It is not your actual response, it's a generator. So now if you want to get the information from this generator, you have to do for chunk in.
Response.
Then print chunk dot content. So whatever you see right end equals to.
What is this codes? This means let's suppose you have 10 numbers 123456. If you run a loop which is for I in range.

Tirth 1:25:24
M.
Hmm.

Tarun Jain 1:25:39
10.
If I do print I this will print everything in one single what you call in next line. But if you want this to be printed in one line you just use end.
In space.
Is this clear? So whatever response I get, I don't want every tokens to be in next line. I want to be like it's typing something which is in single line. If it sleeps back slash N then go to the next line. So now I'll just run this.

Tirth 1:26:00
Yeah.

Tarun Jain 1:26:25
OK, this is collab. What do we have to do? But still, if you see it's generating 1 token at a time, do you see that the difference will be much better if you run this in VS code.

Tirth 1:26:29
Yeah.
Mm.

Tarun Jain 1:26:37
The seconds if you see it's still the same, but The thing is it will start generating one word at a time.

Tirth 1:26:43
Mm.

Tarun Jain 1:26:46
So try this example on, I mean VS code, the LLM stream right this on VS code.
So we covered invoke and as well as we covered stream.
Stream is like 1 token at a time and the data type is a generator.

Tirth 1:27:10
Mm.

Tarun Jain 1:27:17
Till here is it clear?

RamKrishna Bhatt 1:27:21
Yeah, for invoke my compiler is still running in a.

Tarun Jain 1:27:27
How much second has it been? You will see that seconds here. If you notice here, can you see the time? There you will see that second.

RamKrishna Bhatt 1:27:32
Yeah, yeah, it's a 2 minute. It's a 2 minute, 55 seconds.

Tarun Jain 1:27:38
OK, what question did you ask?

RamKrishna Bhatt 1:27:41
The same same which we have. What is the not dating texture was in where where it is located. So question was same.

Tarun Jain 1:27:49

Model is Gemini.

RamKrishna Bhatt 1:27:52

Yeah, Gemini 2.5 Pro and I have, but I have just generated Google AI Studio keys and provide the key to the sender. Sure, sure.

Tarun Jain 1:28:00

But can you stop it and run it again?

So do one more thing LLM dot API key.

Select friends.

OK, it's hitting it.

So print this line and see if you can see your keys or not.

RamKrishna Bhatt 1:28:35

Sure.

Yeah, I'm able to see that.

Tirth 1:28:48

I think you might just need to restart the session, but I'm try restart the session and start from the very start.

RamKrishna Bhatt 1:28:57

OK, like I have already restarted at the time when Tarun mentioned, but I will try again soon.

But.

Tarun Jain 1:29:05

OK, so now let's come to the prompt template. Does anyone remember the syntax for prompt template?

Tirth 1:29:13

List of tables.

Tarun Jain 1:29:15
In the sense, how do we import it? Do you remember the import statement?
From the same.

Tirth 1:29:22
From chat.

Tarun Jain 1:29:24
Core dot prompts.

Tirth 1:29:27
Oh no, it was something for prompt chat or something.

Tarun Jain 1:29:33
Start longtime.

Tirth 1:29:34
Yes, yes.

Tarun Jain 1:29:36
And then what we are supposed to do, we just have to define a wrong template.

Tirth 1:29:39
OK.

Tarun Jain 1:29:46
That prompt template.

Tirth 1:29:50
From message.
No.

Tarun Jain 1:29:53
What is the first data type list?

Tirth 1:29:56

Message list list of tuples. Hmm.

Tarun Jain 1:29:58

Message is just equals to list of.

Tirth 1:30:01

All this tuples, so we will have system tuple and yes.

Tarun Jain 1:30:04

So.

Do you have your system from?

I will define system from Tabo.

And then another number.

Tirth 1:30:15

We'll have any user or human or whatever we want to type, yeah.

Tarun Jain 1:30:19

This user and then you have human prompt. So human prompt I've already defined.

Tirth 1:30:26

Ready.

Tarun Jain 1:30:27

Human prompt is query, so we'll just write query system prompt.

Let's see if it controls or not. You are an expert medical assistant.

8.

You only answer the user query if it's related to medical or drugs.

And the user query is not from the medical domain, just say.

Not relevant.

If the above instructions is not followed, you are fired.

Then what is the next step?

Tirth 1:31:32
Then we have to do prompt equal to prompt template to LLM with pipeline.

Tarun Jain 1:31:37
We define that chain.

Tirth 1:31:41
Yeah, yeah.

Tarun Jain 1:31:41
Wrong template.
Then open LLM.

Tirth 1:31:48
And then we do change dot string.

Tarun Jain 1:31:50
For some in saying dot stream, do I have any input variable in query?

Tirth 1:31:56
Mm.
Not enough now.

Tarun Jain 1:32:00
I don't have, so I don't have to give any. What do you call? If in case I have any input variable, what is the syntax?

Tirth 1:32:08
It is written input equal to dictionary, so input equal to.

Tarun Jain 1:32:13
So whatever input variable you use, you have to define that. Let's suppose I define portion. It should be that particular variable. So let's suppose I define.
Question like this.

Tirth 1:32:35
Ready.

Tarun Jain 1:32:37
Ready.

Tirth 1:32:38
M.

Tarun Jain 1:32:39
No.
Chunk dot content and slash.
Not relevant.

Tirth 1:32:52
Hey.

Tarun Jain 1:32:58
Alas, tell me.
About para.
Set the wall and when to use it.
Is it 650 MG or?

Tirth 1:33:16
350MG you will get 25350650 all good.

Tarun Jain 1:33:27
OK, it took more than 10 seconds. That means it will send it the response. It won't say not relevant.
But the stream effect is very slow in collapse.
OK.
Did everyone follow how to use from template? This is same as what we did in the NLM experiment.

Tirth 1:33:54

The input.

Yeah, but we have to type it down. Just give me a second.

Tarun Jain 1:34:02

Is it visible?

Tirth 1:34:03

It is visible.

Hey.

Tarun Jain 1:34:09

So instead of invoke, we use invoke last time. We'll just use stream.

And I'll do one more thing. Whatever this slides is there, right? Uh.

This PDF loaders I will attach this slides link.

In this notebook.

And I will push this particular code in our GitHub repo that we are maintaining Python AI workshop because here I've also added some information that we will cover unstructured IO during multimodal drive.

And why do you use metadata? So I'll just download this and push it there.

Let me know if it is done.

Tirth 1:36:53

Yeah.

Tarun Jain 1:36:55

Oh, everyone also is.

Hello.

Tirth 1:37:01

Yeah.

Tarun Jain 1:37:02

OK, so OK, we are done with the time, but I'll do one task. What you can do is.
Tomorrow when you come, define one parenty class.

Tirth 1:37:14
M.

Tarun Jain 1:37:16
Where you need at least three keys from LLM.
OK, so I'll give one example. I want to define up identity class of FAQs. So what will I have for FAQs?

Tirth 1:37:32
Sure.

Tarun Jain 1:37:34
I'll have question, I will answer and then I'll have something called a justification.

Tirth 1:37:42
OK.

Tarun Jain 1:37:43
So this is a identity class for FAQs just like this. What you guys can have is you can have your own.
One parenting class.
Which will have three keys.
And for each keys you have to define a field and a proper description.

Tirth 1:38:03
OK.

Tarun Jain 1:38:04
So this is something that you can have in place and then what we will do is we will use the same class tomorrow when we work with structure output parsers.
So tomorrow we'll complete structure output parser, then tools, embeddings, creation, vector DB. So once we complete these four things, tomorrow probably we

can start with drag as well. Embedding creation is already done, so I'll just remove this.
Tool calling and vector database is something that we will do.

Tirth 1:38:35
OK.
But we have to still inject the we we still have to inject the embeddings that we have created in the context.

Tarun Jain 1:38:48
Uh, that we can cover under vector database itself.

Tirth 1:38:51
OK.

Tarun Jain 1:38:53
Or let me keep that why is in clearing world.

Tirth 1:38:57
M.

Tarun Jain 1:39:03
And I will just add one line here. Let's try to install fast embed.
And first embed should be fine. If not, we can also use Gemini's embedding and for Gemini embeddings we only need this line.

Tirth 1:39:17
OK.
OK.

Hardip Patel 1:39:22
Are you talking about adding those data that we added in chunk?

Tirth 1:39:29

Yes, yes. So I just want a relevant answer to that. I where do we add that? Let's so about that.

Tarun Jain 1:39:39
Uh, what relevant data?

Tirth 1:39:42
So we need the loader one, loader two, loader three and we want to output related to that only.

Hardip Patel 1:39:42
Uh, so.

Tarun Jain 1:39:45
Huh.
OK, OK. You mean you want to build rack pipeline, that one, right?

Tirth 1:39:53
Yeah, like uh, with line chain, we are not using that as a context yet. We are using the direct context one shot with the LLM.
But we want to pass the query and the context, the context that we would find from the vector search.

Hardip Patel 1:40:09
I think we are using, yeah.

Tirth 1:40:16
Meet if you uh.

Tarun Jain 1:40:16
Uh no, I didn't get the context.

Hardip Patel 1:40:18
When when we are piping the LLM, I think that's when we are using the context, no?

Tirth 1:40:19
OK, so.

Tarun Jain 1:40:21
Yeah.
For argument or.

Hardip Patel 1:40:31
I mean whatever code not might be loaded.

Tirth 1:40:31
So we if you scroll above, yeah, if you scroll above, we loaded the PDFs, right?

Tarun Jain 1:40:40
Yeah, this one.

Tirth 1:40:40
Yes. Oh yeah, the PDF in the website. Now where do we add this as a context or queries to the LLM?

Tarun Jain 1:40:45
Yeah.
OK, that will add it here in the vector DB first.

Tirth 1:40:51
OK, OK.

Tarun Jain 1:40:53
So Vector DB I will show you 2 approaches.
So the first approach is from line 10 itself.
Langchain has their own vector databases.
And methods they are defined. So these methods are defined by Langston itself. But if you want to have your own configuration for own configuration.

Tirth 1:41:15
OK.

Tarun Jain 1:41:23
We will index it directly using the vector database framework.

Tirth 1:41:30
OK.

Tarun Jain 1:41:33
Instead of.
Relying on.
So what are this configuration? Let's suppose you have.
Large PDFS, which is five to six PDFS that may take too much of time. That will take memory. That will take RAM. That is where we'll be using binary quantization.
And what are the other quantization? I mean what are the other configuration we have? I did search.
And then if you want to do metadata filtering in payload.
Metadata filtering in payload that is based on specific source only you want to extract the information. During that time you might need metadata filtering from payload. So this is what we will be doing by defining our own configuration. So these are two approaches.

Tirth 1:42:28
OK.

Tarun Jain 1:42:44
And what I will do is whatever I loaded from web-based loader right? We will build chatbot for this data.

Tirth 1:42:50
M.

Tarun Jain 1:42:52

Where is it?
I will build chat for for this particular data and since you already have this PDF right?

Tirth 1:43:02
OK.

Tarun Jain 1:43:03
Career leveling you guys can build.

Tirth 1:43:04
Oh.

Tarun Jain 1:43:07
Drag for that particular PDF data. So then what you can do is you'll have two code available for you, one for website, one for PDF.

Tirth 1:43:10
OK.
OK.

Tarun Jain 1:43:17
But the same thing is once you have chunks right, once you have this particular line, the remaining code is same. The only code which is differing is before chunk. Before chunk I'm using web-based loader. You guys will be using PDF loader. After chunks all the information is same.

Tirth 1:43:25
Hmm.

Tarun Jain 1:43:35
Or all the code lines, whatever you write, it will be same.

Tirth 1:43:40
OK.

Tarun Jain 1:43:44
OK, that's it, I guess. Let's just make sure we have one pedantic class tomorrow, so it's hardly 4 lines of code.

Tirth 1:43:49
OK, OK, sounds good.

Tarun Jain 1:43:52
Not four lines, it will be 5 lines. One is import statement, then one is class and then three keys 5 lines.

Tirth 1:43:56
But.
Thanks, Devin.

Tarun Jain 1:44:01
Yeah.

Tirth 1:44:04
Thank you so much all.

RamKrishna Bhatt 1:44:05
Thank you. Bye, bye.

Ajay Patel stopped transcription