Python and AI Power-Up Program Offline Class-20250911_113127-Meeting Recording

September 11, 2025, 6:01AM

1h 26m 22s

Tirth started transcription

Tarun Jain 0:13
OK, I guess everyone now is here, right?

Tirth 0:19
Ajay will join late and we'll to start recording which we already have.

Tarun Jain 0:24
OK, so I'll share my screen.
So asking.
Uh, I hope this screen is visible, right?

Hardip Patel 0:39
Yes.

Tirth 0:40
Yes, it is, yes.

Tarun Jain 0:42
So what we'll do is we'll continue with the vector database itself. So main focus today will be this thing. We'll try to have filtering condition. Then let's suppose you have metadata. So how you can use metadata and then get the relevant documents. So this will also help in the.
Project which you will be building in the RFP. So I'll tell you how you can use it and what are the different use cases you can build. OK, so coming back to the same code base. So what we can do is last time what we did was we installed Langchain, Langchain community by PDFM.
Then fast embed and then also quadrant client right? So what we can do is since fast embed, if you remember last time we didn't use langchain, we directly used from fast

embed import text embeddings.
And then we had sparse text embeddings, right? So what we will do is we don't have to install linechain and linechain community. Since we are doing inference, we can directly install fast embed, then quadrant client, then line graph.
Uh, blanks and this is for prompts and then the LLM so we can just install this part for the inference.
Since we have already saved the data in the fast embed, so we will continue with the hybrid search.
So I read search we have the collection and right we will reuse the same collection.
So these are the pollens. You can use the same collab or if in case you have VS code then probably these are already installed in VS code.
So I'll just paste it here.
And let me also open this.
Once this is installed, probably what I'll do is I'll Scroll down. Here we need to load this embedding model. So what I'll do is I will run this import statements. So from quadrant client, import quadrant client, then comma models and then all this configuration. This is not required now. Why? Because we have already saved.
Data inside vector database. So distance vector params and sparse vector params is to create the collection.
And how are we supposed to define? We are supposed to use the line then dot create collection. Inside this you have to define your vector configuration.
And sparse configuration.
And if you notice the syntax once you have collection name.
You have to define this keyword. This keyword is very important because when you are using query points and if you want to return the relevant documents, you have to use a parameter called using right? After using you have to tell these keywords. So as soon as you define vector config what you need to do is you have.
To define dense and then vector params. Same goes for sparse vectors where I'm defining sparse. So I hope this part was clear. Then binary quantization we have already repeated two times. So what I'll do is I will just run these models. One is dense embedding model.
Where the where we are using text embeddings.
The.
Did anyone try with land graph the inference part?

Hardip Patel 4:58
Not yet.

Tarun Jain 5:00
OK, uh, we'll do that now.
So what you can do is after you run the installation, just run this part where we had imported quadrant client and all and then we should also have the client. This is API, then URL and the collection name is same which is hybrid.
So client collection name then two models. We don't have to create collection.
And here what we can do is we can directly use the query from the user draw inference. So this is the inference where we have user query.
So I'm running this query then dense vector. So what is the next step? As soon as you have query in the rack pipeline, what are we supposed to do? We need to convert that into embeddings and once you convert into embeddings you have to check that into Vector DB.
And vector DB will return the top K chunks.
All right, so if I come back here, I'm just converting the user query into vectors. And what are the major things you have to notice when you're using prefetch? For every time you're using sparse vectors, there is a parameter called as object. You need to use this particular function.
And this is a keyword argument, so I hope you understood why I gave double asterisk.
And when it comes to hybrid search, we need to have the number of relevant documents from vector search and also from keyword search. And here from dense, which is your vector search, I'll get 10 documents.
For sparse I will get 10 documents and then based on the scores I only need top three so that is defined inside query points. So query points is nothing but search. Search and return the relevant document.
So I'll just run this vector then prefetch top K and here last time what we did was we used context. So just make that context as context one because I'll show you the filter and MMR reordering as well.
So here what you can do is instead of context one, I mean instead of context, just make it context one. Here I'm defining collection name, then prefetch. Prefetch has to be defined only when you are using hybrid search. If you are not using hybrid

search, you can skip the prefetch.
And then the query is dense vectors and I'm using dense. So this keyword needs to match with what you have used in collection and then payload. Payload. If it is false, what will happen if I make this false?
So suppose I make this.

Tirth 7:59
We would only get the metadata. We won't get the text content. We won't get it.

Tarun Jain 8:04
Correct. So now if I run this and now if I run context 1.0 payload source, this will be error. Why? Because my context .0 payload is empty.
It is none. So now if I make this true.
I'm getting the context, so I'll just keep it metadata.
Payload source, sorry.
Payload itself is a metadata. So now if you notice I have total 3. So what is the length of context one dot points? It should be 3.
And for every single source if you see from first source it was using SARS development and then in the second source it is using about us and then in the third source it is using the home page. So now just imagine you have a user query.
So you have user query.
And you have 10 documents.
10 PDF files.
And if you remember most of the platforms that you have right, whether it is AI Studio or Gemini, they have flexibility of changing certain parameters. So what you can do is whenever you're building certain drag application, you can give flexibility for the internal team or users.
To pick from which source it needs to answer. So this is 1 filtering technique. If not, you can also use other parameters like keywords and all. I'll come to the keywords part, but now let's imagine you have user query. So from this user query.
Which document it is supposed to use to get the context?
If you don't use filtering it will do search and it will get you the results. So now what happens in filtering is you will apply a filter and you will tell only use document three.
So now when you are getting the context the source of all.
Top three will be document three you understood. So here it can be document one,

it can be document four, it can be document five. But here as soon as you apply a filter you're telling hey I know this query is coming from document three, so you only have to get the context from document three.
That is the filter.
Are you understood the use case what we're trying to do?
We are telling from which document or which source this information needs to come. So this is 1 use case. So now let's talk about second use case. I don't know why this is not refreshing.

Tirth 10:40
Yeah.

Tarun Jain 10:51
So if you notice in the Uber thing they have their policy documents and after they have policy documents they are trying to extract the custom metadata. In this custom metadata there are two things.
One is let's suppose you have 10 chunks.
And you have one document.
Now for every single chunk, what you need to do is one is you have source, then you have page number.
Then you have title. Then what you need to do is you need to add one more additional.
Key which is document summary.
And then you have something called as chunk FAQ and then you have chunk keywords.
So this will repeat for every single chunk. But what you need to do is this document summary, whatever you have will remain same. This will remain same across all the chunks.
But whereas chunk FAQ will be unique for every chunk.
This will also be unique for every chunk.
But the document summary same only chunk FAQ and chunk keywords will be different. Now I'll tell you why this is important. So for the use cases like business requirement document which is VRD, PRD and also RFP, let's suppose.
You have an RFP just to give you some information about what is RFP. So basically Jabbia Kogu's project you will get right. Let's suppose you have a project and this is

client requirements. You will have a document with client requirements.
And you'll also have certain questions that you need to fill.
Questions that needs to be filled.
And once you questions you have to submit the proposal. So you have document which has the client requirements, then you have questions. So this questions you have to fill it every single question based on your internal documents and your past experiences and then you have to submit a proposal and in your knowledge base.
Inside your knowledge base you will have certain internal documents.
So now this is there in your vector database.
So now just imagine whatever your new project is there, it belongs to manufacturing domain.
OK, I'm just taking an example. You can pick any domain that you need, but I'll safely take retail because you have worked in the retail domain. So you have document. So this document, whatever new client requirement is there, it is in retail domain. Now whatever knowledge base that you have, it is.
In some other domain, let's suppose it is in finance domain.
So now whenever you extract any information from your knowledge base, you will get the finance related chunks. But there are also chances in your internal documents you also have retail data.
So now what will happen is every single chunk that you have, it will tell from which particular domain does this source belongs to or the content page content. So you have page content right?
So now what will happen? When you pass this chunk keywords, you're generating it from LLM. So all these three things, document summary, chunk FAQs and chunk keywords, you're generating it from LLM and that LLM is using page content as the context.
So now what will happen is this chunk keywords will have certain unique keywords. This can be related to retail, this can be related to finance. So now when you are creating this proposal, you only want those chunks which belongs to retail. If it doesn't belongs to retail, you can just remove it.
Does this make sense? So whatever is related to finance, that is not relevant even though you have both, but that's not relevant to your existing document. So you need to provide certain filtering technique. So this is 1 use case. So this use case is very common when it comes to most of the report generation use cases where filtering is very.

Very important. So those filtering can be only applied when you have some understanding of what domain expertise you need from LLM to be extracted and every single time you will have your own internal documents or it can be other documents as well.
Does this make sense?

Hardip Patel 15:42
Yes.

Tirth 15:43
It.

Tarun Jain 15:43
So this is just use cases that you can build around filtering.
And one of the example is this folks itself. I'll also show you the code how actually it looks like.
So if you see or can you see this code generate document summary?
So you have generate document summary, then you have generate some data and now if I see where it is extracted. So if you see you have generate document summary. So from where am I extracting this full text? I'm extracting the full text from.

Hardip Patel 16:06
Yes.

Tarun Jain 16:21
Page content. Same goes for chunk metadata. So what am I trying to do? I want to generate metadata for each chunk and for each chunk what is my input? The input is page content and once I get that page content, what am I supposed to do?
I need to generate two or three relevant keywords. I mean two or three relevant FAQs which is frequently asked questions and also five to seven key unique topics or keywords. So this topic keywords is relevant based on the use cases, right? So this is just additional.
Metadata that you can add which can be later be used for filtering technique like if it belongs to this particular area.

Tirth 17:01
Oh, like that.

Tarun Jain 17:04
How?

Tirth 17:04
So Karan, we are asking only for generating this summary or FAQ pairs or topics and keywords from the content that we are providing of the chunk.

Tarun Jain 17:17
Oh, can you repeat? There was a little bit blur in your voice.

Tirth 17:18
So so we are asking the LLMS themselves for creating this, you know, summary and junk data.

Tarun Jain 17:27
Correct, yeah.

Tirth 17:29
Wouldn't that be a lot? Because if I have a big PDF of let's say 300 pages and then it has to create chunk size of and let's say it is creating 1000 chunks, then put it 2000 calls for element.
You.

Tarun Jain 17:44
Yeah, it will do the thousand calls.

Tirth 17:46
Don't get me over expensive, just trying to understand.

Tarun Jain 17:52
So here there are again multiple ways we can do so if you notice this diagram so they

have mentioned large and small. So if you see some places they have used small LLM, some places they have used large LLM. So since this is happening on the document offline process, usually this will take too much of time to.
To save it in Vector DB. Now coming to COSH, these folks are using large LLM.
Probably based on your requirements, you can keep the small large language model. Small in the sense GPT 4 or mini. If not, you can use any open source LLMS.
Usually for custom metadata collection, most of the time what happens is people use LLMS. If not many people use LLMS, you can use rule based approach like if you know you have the logic for which you can create the metadata, you can write the logic for that.
But again, this logic depends on you how you define this logic.

Tirth 18:53
Understood. Understood.

Tarun Jain 18:53
Since here I was getting the keywords, I wanted the keywords. The best option was to use an LLM, right? So you can use two different LLMS. One is your main LLM, one more is a small LLM. So small LLM can be anything.
So they're using small LLM in some of the few use cases. I mean some components. Here it is small, here it is small, here it is small.

Tirth 19:20
OK.

Tarun Jain 19:21
So whichever model you see right, which has 4 billion parameters like uh, there is Microsoft's.
Pi 3.5 This is very small model and hardly if you do thousand calls right, you won't even spend more than $5.
So metadata creation.
Is a offline mode.
Offline mode is like you won't let user to do it. You will save the vector database or you'll create the knowledge document on your end. And if in case you're giving user to create their own knowledge base, what you can do is you have a page.

User uploads the document.
And then what you can do is you can show a message saying that once the indexing is done, you will send an automatic e-mail that hey we have saved your index.

Tirth 20:25
That's good.

Tarun Jain 20:25
But typically not many people use saving the document live. Usually this happens in the offline mode.
I guess even they are doing it in the offline mode.
So if you see two key components, one is offline document processing and then real-time answer generation. So this offline document processing is this part from documents, then metadata and then chunking and embedding model and then save it in a vector DB.
So till saving in vector DB, everything is happening in the offline mode.
And filtering is applied here. If you see a metadata filters which is happening on the answer generation side which is in real time.

Tirth 21:14
Yes.

Tarun Jain 21:14
Is this clear? So what we have done so far, we didn't do chunking yet. I mean, we didn't do custom metadata filtering yet. And what didn't we didn't do? We didn't even do this part, which is the metadata filtering.

Hardip Patel 21:18
Yes.

Tarun Jain 21:30
But if you notice the other blocks.

Hardip Patel 21:30
It was.

Tarun Jain 21:33

After user query is extracted here if you notice the retrieval process you have vector search and then you have beyond 25 search which you have already done.
It is this part.
So you have vector search and you have keyword search. So for sparse what is the model we are using? We are using beyond 25.

Tirth 21:54

BF25BF.

Tarun Jain 21:56

So this is the same thing what even Uber is using and then you have your retrieve chunks. Once you have retrieve chunks, post processor is nothing but you use LLM to evaluate and then you have answer generation which will generate the final response.
So I'll show you this process now this one how to do metadata filter.
And it's not just Uber. Most of the people have their filtering techniques.

Tirth 22:21

So we will want to ask SLM or LLM from our query to page to create what kind of metadata or keywords that we want to filter.

Tarun Jain 22:33

Correct. So here what happens in query filtering is let's suppose you ask any user query, right? So I ask some random questions. So this random question is not properly written by the user. So what this small LLM will do is it will optimize your user query and it will see if it can add certain metadata or not.
So this is happening via prompting again, prompting an NLM. You're just improvising your user message with metadata. Like what are the possible metadata that you can use for this user query?

Tirth 23:00

Nothing.

And what is feature store like does it use? It is saying over here that it uses feature store to generate the metadata and when we are having this you know processing.

Tarun Jain 23:15
Each sun story is nothing but peered.

Tirth 23:18
OK.

Tarun Jain 23:19
Feature store is nothing but payload here. So usually store is there, right? Let's suppose you have tables.

Tirth 23:22
Yes.

Tarun Jain 23:28
Um, where is it? Suppose you have a PDF.
And this PDF has images.
It has tables and this images. What am I saying? It's like it has some text on the image.
So when you're using parsers, most of the time what happens is you are only extracting the image path. So whatever the figure is there, right? Let's suppose you have image below image, you have pic caption.
So when you're using parsers, you're only extracting the caption. You're not extracting what is within the image, right? And then you have tables and then you have page content.
And then you have metadata.
So what these folks are doing is they created their own custom metadata, they added in the payload. So whatever image is there, you create certain artifact and you save that in the feature store. So feature store is just the payload and whatever vector store is there.
Vector store is nothing but your embeddings.
Which is your page content?
And if I show that in our code.

Where is our code? So can you see this part? This is vector store and this is feature store.
Oh, you understood. So here you can also save images. So here I have document, I have source. Here you can also add table summaries.

Tirth 25:06
OK.

TJ **Tarun Jain** 25:08
Table summary then whatever that chunk is and then you will have what else is usually added. Table summary is there, images is there. Images is not chunk rather it is file path basically. So in their blog they're not yet using OCR. So if I show you the. Set.
So as of now they're using Google Docs plugin. So do Google Docs plugin only supports textual content. So they're yet to extend the support for multimodal. Multimodal is basically images, audios and all. So as of now they're not doing the images, but usually images is saved in.
Uh, what do you call payload?

Hardip Patel 25:49
I thought.

TJ **Tarun Jain** 25:51
Where is that here?
And in order to do that, there is something called as unstructured IO. Did I mention this keyword before?

Hardip Patel 26:01
OK.

TJ **Tarun Jain** 26:07
Structured IO so unstructured IO has something called as partition PDF.
So what happens in partition PDF is it creates a separate folder for images. It will create a separate folder for. So if you see you have partition image, it will create separate for image, it will create separate for text, it will create separate for tables.

And then you can save each one of them in a payload, which is a feature store.
So if you see there is an example directory. So this directory will have tables, images and text data. So it's a folder that it will create.
Does this answer your question?

Tirth 26:53
It does. It does.

TJ **Tarun Jain** 26:54
So this will be very helpful in 90% of the use cases having payload. Payload is very important.
OK, so I'm not run this cell because this cell only needs to be done when you are saving your vector database and then using upsert and here we are doing inference user query converting into vectors, prefetch top K and now we have the results.
So now what I will do is I want my source use source as.
I want to use only a voters.
This is the source I want to use.
And now what we can do is we can apply the filter technique where you have to define client dot.
Create it should be payload index, create payload index and here define collection name.
And if you see you have two things, one is field name. Field name is nothing but what is the name that you have defined when you defined your payload. So how many keys we have in payload?
So if I do context one of points.
Of 0 index then payload. What is the data type of this?

Tirth 28:23
Cincinnati.

TJ **Tarun Jain** 28:23
It's a good study.
Right. So now if I do keys.
So these are field name, one is document and one more is source.
So document and source these are field name. So here what will I do? I will just

define field name.
Equals to the source.
Because whatever I'm trying to do here it is from source and now what you can do is just define field.
dot schema equals to models dot.
It should be payload.
Model dot.
OK, where is auto counted?
I need to use this thing payload type schema.
And then keyword.
So I'll repeat what we did. So I need to create a payload index first. So why am I creating this so that I can use?
Filtering on the metadata. So when you are applying filtering on the metadata, you need to define. You need to define your what you call field name.
So field name is basically your payload E values. In our case it is.
For example.
It is document and source. So this is your field name and then what you're supposed to do is you need to match it.
Based on the keyword. So the keyword is nothing but it should match exactly same as what you have defined here. If I do Virat Kohli, let's suppose I have use source two and I defined it as Virat Kohli.
So whatever relevant documents it is returning, it should be none. If I use this particular variable and we'll test it with both this use source till here is it clear? We are just defining certain logic on what you have to, I mean which particular key needs to be exposed for filtering.
And the key that we are exposing is source. So if you have chunky words, let's suppose you will follow the pattern of how Uber did it and how many values they have. So in that particular metadata.
Somewhere I've added that in here.
So how many field name do we have here? One which is source, then you have page #2, then you have document summary 3, then chunk ID, chunk FAQ and chunk keywords. So these are all keywords. Now let's imagine you want chunk keywords to be exposed for filtering. What will be your logic?
Client create.
Payload index.

Here you'll define the collection name.
As collection new.
And then field name will be your chunk keywords and then field schema SM. So these are the three parameters you have to define.
Till here is it done? You can just define these two things, one simple variable and then expose the key. I'm telling key because field name and key same and I'm just referring to dictionary keys so that key source and then the schema which is keyword.

32:43
Yes.

Tarun Jain 33:00
Let me know if it is done.
I will remove this. This was just to showcase one more example.
I hope everyone understood this thing, right?

33:20
Yes.

Tarun Jain 33:21
OK.
So now what we can do is scroll up, copy this entire thing, whatever you defined in context one, copy this entire thing, paste it here and instead of context one, make it context 2. Collection name will be same, prefetch will be same, query will be same. Using is also same, load index is same, limit is same. The only new thing what we need to do now is I will define a query filter.
Query filter is a parameter and what I will do is I will define that query filter outside. And then just assign this query filter here.
Just copy these two things. One is query filter and whatever we did earlier for context one, copy the same thing. Just add this new parameter query filter equals to query filter.
Is it done?
Is this done this part? OK, so now whenever you're applying filtering, there are different conditions like one it should match which is must, then you have should

and then you have must not. So all these things has different logic. So what I'll do is I'll just copy this.

Ishan Chavda 34:56
Yeah.

Tirth 34:56
Yes.

Tarun Jain 35:11
Must.
Filter important. So if you see you have must, you have should and you have must not. So basically what you're supposed to do is you have to define models dot filter and then define this keyword. This keyword should be match or match not or it should be should.
So what we'll do is we'll go with match.
So if you see match then I will apply the field condition. In this field condition what am I supposed to do? I need to define a key and once I define the key I have to define the match. So I want you guys to tell me what will be the key now.
What will be the key will be source? Then what will be this line match equals to models dot match value then?

Tirth 35:56
Um value value people to use source.

Tarun Jain 36:01
Correct. So what I'll do is I'll just copy this.
I will come back here and here I just have to define.
Models dot filter.
First I need to define models dot filter and inside filter I will copy this.
Filter inside filter you have must must. Then you are starting with the square bracket and then models dot field condition. This should be source.
And then you need to have a match condition match equals to models dot match value. What is key is use source?

Hardip Patel 37:03
OK, good.

Tarun Jain 37:06
And then this is closed. Then I need to close that one and this is closed. Huh. So filter is closed. Then much is in square bracket that is closed and filter condition is in parenthesis that is also closed. So this is our filter condition.

Hardip Patel 37:11
M.
We've got some.

Tarun Jain 37:29
And now this filter, this query filter. I'm using it here. Now you just run this.
And now I'll just copy these lines, whatever earlier I had. So for all the three that I have right here, if you see 012, it is different. Here I have size development, here I have abort, here I have the home page.

Hardip Patel 37:40
OK.

Tarun Jain 37:51
Now if I print this line context 2 of 0.
It is a voters context 2.1. It is a voters context 2.3. It is a voters.
Sorry, it should be too.
So now let's talk about edge cases. What if I only got 2 relevant documents from about us? So what you can do is during that time you can just take the zeroth index, zero or first index. So what you usually have to do is you have to define a if condition.
If none, then what you can do is you can try to concatenate them and provide it to your land graph node. You understood if in case it is empty.
You can copy this link.

Tirth 38:46
But wouldn't the reparative payload of the document that it would get done?

Tarun Jain 38:54
So for which one?

Tirth 38:55
So when we are doing this, when we are doing query filter and now it is giving us three back from same about us. So would it have document repeated or?

Tarun Jain 38:59
Yeah.
Correct.
Yeah, this is right answer. So that is one negative case of using filtering. So when you applied any source, right, let's suppose in this use case it is fine, but when it comes to legal judgments, you have legal judgments, you will have around 50 documents.

Tirth 39:07
It won't be repeated.

Tarun Jain 39:23
So the domain expert knows whatever question he has asked, he has asked from the document 30.
So now what you will do is you will use your source and here it will be the path of the document and now whatever search parameters are getting right, search relevant document, it will be from there and there are high chances you will find repeated documents and not just that the second edge case is.
Whatever chunk you have here, this might be irrelevant.
You understood the edge case. One is repetition and 2nd is irrelevant. During that time what you will do is you will reorder.
So what happens in reordering is you will use relevance plus diversity.
So this is where usually the ranking comes in.
Where is it?
So once you do reordering right after reordering you also have reranking. So

reranking what will happen is it will base it will depend on sports.
This.
You understood. So the two edge cases are one is repetition.
And 2nd is irrelevant.
So in order to solve the irrelevant, what you can do is based on scores, we filter it out.
And for repetition, what we do is instead of just relying on relevance, we will also do relevance plus.
Diversity.
And in pretty much use cases, you will not have just four or five web pages. You will definitely take 10 web pages. And when you take 10 web pages, if there is more content in it, during that time you can apply filtering.
And these are the two common edge cases when you're doing filtering. One is repetition and one more is irrelevancy.
Is it clear?

Tirth 41:55
Yes, it is.

TJ

Tarun Jain 41:57
These two issues will be there.

Tirth 41:59
OK.

TJ

Tarun Jain 42:05
But we can tackle this. I'll show the logic for the repetition one, the relevant one. I'll show you the logic on how you can filter it out, but this is pretty easy. So if you print points to it, let's suppose.
I will just print points to here you should have scores.
So if this score is less than 65%, don't even take that context.
And if you notice the.

Tirth 42:37

So that would that would solve the irrelevant. That would solve the irrelevant problem.

Tarun Jain 42:42
Correct. Here you use filter based on scores, so observe this also if you do context 1.

Tirth 42:43
OK.
Um.

Tarun Jain 42:50
Context once.
Of points 0 index will always be more, so I'll use .1 then score.
Let me run this below.

Hardip Patel 43:09
Point spice find S.

Tarun Jain 43:16
Context to points. OK, it's a bit low. 0th index, 0th index will be same I guess because it came from same document. OK, it's still low if I do the second index.
So you saw when it comes to context one, since it was using all the documents, it has scores which are more than 80. But if you look at these scores for context two, it is very lower compared to the context one.
But still it is more than 65%. Usually we keep threshold to be 65, which is 0.65. We convert this into 100 and then we filter it it as. If it is less than 65 then we won't take that document.
And now what you can do is instead of use source, make it use source two which was Virat Kohli.
Dimple on this.
And now if I run this.
Context to.
It's empty.
Everyone understood this logic.
These are the only two cells, query filter and then context tool.

And not all the use cases you will use filtering.
If you're building question and answer chatbot, in question and answer chatbot you will not use filtering. Filtering for RFP for report generation filtering will be there.
So here also in their use case they have policy documents. This knowledge documents is nothing but their policy documents.
So for Q&A chatbot.
QA chatbot or support service chatbots.
Usually filtering won't be applied and this is again this one is arguable. Let's suppose in your knowledge base.
Suppose you have knowledge base.
And what you have done is you saved your finance document. You have saved your. Take documents.
And you have also saved something related to product documents.
And what else do we have? Finance, tech, products, sales, documents, finance and sales are same or is it different?
OK, so let's suppose you have these four documents. Now you ask any question and this question is overlapping between tech document and product document and you're building a customer service chatbot. So usually if I was supposed to approach the solution, I would rather use routing.
Rather than using filtering.
So this is what chatbots I'm saying. For chatbots, basically filtering is not used.
You understood the problem statement, but when it comes to report generations like you're editing documents to documents.

Tirth 46:48
What is the routing?

TJ

Tarun Jain 46:53
Document creation or report generation.
Where your knowledge documents, whatever internal documents are there, it's same, but whatever inputs you add might be different for different industries. So during those edge cases, filtering is very important.

Tirth 47:21
What is routing? You mentioned that you use routing for it.

Tarun Jain 47:24

The thing is basically let's suppose you have user query.
After user query, your first layer will be intent classification.
So basically here what you will do is you will decide whether it should go to the tech team or it should go to finance team.
Or it should go to sales team?
Or it should go to what was one more product. So based on this what it will do is it will do classification. So classification it will tell hey it should go to tech, it should go to finance. So basically this will be 0 or one or two or three.
Right now if it is zero, it is for tech team. Then we use rag. Once it is routed to rag, I mean once it is routed to tech team, I will use that. Didn't I show this diagram before for routing?
For Landgraf, when we were discussing, I showed it this thing. Do you remember this?

Hardip Patel 48:30

Yes.

Tarun Jain 48:33

So we had parallelism, we had routing. So here if you see for routing, whenever you have any user query you have decision making LLM and this dotted marks are very important. It won't go for everything, it will only go to the direct tech team. So this is routing basically which is an agentic application.
Routing is an agent, basically.
But when it comes to report generation, blindly use filtering because it is very useful and for that have columns like summary.
Then what you call keywords.
And FAQs.
And this is not like what these folks did by their end. They also copied from some research paper.
I wanted to show our product, but uh, I'm not yet concerned about timing whether I have the.
Thing that I can show or not?
I'll probably take the consent and then I'll show this, but once you guys have your

project ready, we do have one RFP project that we had built, but I'll show this once you guys build. I also want to check if I can show that publicly or not.
But is this clear? These two?

Hardip Patel 49:59
Yes.

TJ **Tarun Jain** 50:00
What?
Once you guys build RFP right, probably I'll check with the team if I'm allowed to show that or not and then I'll show some workflow on how usually we build it and you guys will also build something similar and it's very useful, very useful POC.
Till here is it done the context 2 and the filtering.
Anyone is getting any error or has any questions?
Uh, hello.

Hardip Patel 50:36
No.

Tirth 50:36
Not getting any error from my side.

TJ **Tarun Jain** 50:38
OK, So what we can do is I can copy this context to.
And paste it here and instead of context 2, just make it context 3.
So if you want query filter, you can keep it. If not, you can also comment it.
OK, so I'll just show one thing medium.
OK, so let's suppose now you have any user query. Basically what is happening is till now whatever search we have used, it is like OK, you ask a question. Once you ask a question, whatever is relevant based on vector search, you're getting the relevant query, right? But in some of the use cases like recommendation system.
So far what are the use cases we have explored? One is the QA thing. So QA can come in your chat bots or it can be your customer service assistance.
Customer service assistance.
Then we had something on the document side which is BRD. Then you have PRDS,

then you have RFP.
And then we have recommendation systems.
And recommendation system also we have already worked where we saw TFIDF for movie related data set.
So when it comes to recommendation system, most of the time what happens you can't just rely it on relevance, you also need some kind of diversity. So here I have one simple use case. If you see I want to know what are the Indian foods for first time visitors. If you use relevant you'll only get.
Those which were repeated most of the time, which is curry. But if you need diversity, what you will do, you will look at other available options as well. So if you need both relevance and diversity, this is where we have MMR, which is maximum marginal relevance, which we unknowingly have used multiple times, right? So.
Here if you see there is a parameter called Lambda. So this parameter is very important. If you set Lambda to be one, you won't get any relevance, you will only get diversity and which is not good for our application, just having diversity, right? But if it is very close to 0 then there is no need of using MMR. Also you can directly use similarity. So what we usually do is we define a value which is in between zero and one which is 0.5. So what 0.5 will do is it will use both relevance and as well as diversity.
Is this clear this parameter?
And this is what we need to pass. We need to pass a query vector and once we pass a query vector, we also need to pass candidate. So candidate is nothing but how much options are you supposed to provide. So let's suppose.

Hardip Patel 53:41
They have to be at home.

TJ

Tarun Jain 53:56
I have a user query.
How many documents should I consider which is known as candidates?
Right. And the second thing is the Lambda.
The Lambda is nothing but we define zero to be diverse. Sorry, 0 is relevance.
And one is diverse.
If you define 0.5.
It is mixed of both.

So one is candidates and one more is Lambda. So again, this is from the research paper itself, whatever formula I built.
Candidates documents we already have. Documents is already saved. If you see documents is already selected, then you have query vector. R is not required, we just need Lambda query vector and candidates. So what I'll do is we'll come back to code. Here.
Instead of directly passing dense vectors, cut this out and just define models dot. Nearest. OK, this is very tricky. It should be small query or.
Nearest query and then you can just define nearest.
So whatever you have cut right, which is your query vectors, define it here.
So it was dense vector.
Itis dense vectors.
You understood what we are doing. Instead of directly passing query which was dense vectors, add it inside MMR which is nearest query and then define your MMR equals to models dot.
MMI.
Then diversity is.
0.5.
And candidates.
Candidate limits. I'll keep it 25.
So you just have to update your query. So update your query from dense vectors to models dot nearest query, add your nearest which is your vectors and then define MMR equals to models dot MMR diversity and candidates limit.
So far how did we define MMR? So you define DB which is from line chain. So this was line chain quadrant vector store.
And then you do DB dot maximum marginal relevancy search. So in that maximum marginal relevancy search, it was not directly from MMM, I mean quadrant. They directly implemented this paper logic.
Here what we are doing is we are trying to add the flexibility directly from vector database and this is there again in all the vector databases whether it is VV8 pine cone.
Or quadrant. I'm not yet sure about. OK, even post race has it.
Post SQLMMR.
Hi, it's there MMR. Even Postgres has it. We'll have one session on Postgres.
And then using dense with payload is true, limit is talking. That's it.

And then again you can print context 3.
dot points of 0.
dot score.
1.
Yeah.
So now if you see you're getting a entry which has 77. So this is where it came from diverse.
So this particular query is diverse and I guess this is again relevance.
So now if I make this as 4.
This is again also very less. This also probably might be diverse. So if I don't define this, let's suppose I come back to context 1.
In context one I will define it as four. Probably all the four will be more than 80%. So instead of source I will do score.
It's more than 80.
This is also more than 80.
This is also more than 80.
This is also more than 80, so that means more than 80 is where you are getting relevance. But wherever you're seeing less than 79, these are mainly coming from divers. So what are the use cases where you have to use MMR mainly for creative chat bots where you need?
More understanding of what your product is and I usually use MMR everywhere and second thing is mainly for recommendation systems, but it's OK to use MMR everywhere.
Because obviously you need little bit dense right to get more understanding of your data rather than focusing on just relevance.

Tirth 59:42
Thanks.

Tarun Jain 59:48
Yeah, anyone was asking something.

Tirth 59:50
If you're creating RSP, you don't need the diversity partners in RSP creation that tools that you have shown.

Tarun Jain 59:58

No, we'll need right? So for RFP, let's suppose in your internal documents. So what are you extracting from?
Where did the diagram go?
Where did I mention RFP?
OK, no worries. I'll write it again. So if you remember, one is questions for RFP and one more is document. So this document is nothing but client requirements.

Hardip Patel 1:00:22

You're added in a collab model.

Tarun Jain 1:00:33

And then you have internal documents.
So this internal documents is in your vector DB so.
Your.
Vector DB So now what you're trying to do is let's suppose you have 10 questions.
And I'm picking the first question where you need to look at your previous projects or previous things that you have done from internal document. Do you think for this question you need all relevance?

Tirth 1:01:10

It's from different projection, different documents what we have.

Tarun Jain 1:01:15

So I'll give you one example. Let's suppose in your internal documents and your client requirements, it's related to coding use cases. You want to build a software.
That is related to.
Software that is related to test case generation.
OK, so this is saved in your vector DB. Now like how you built your test case generation, you have the PRD for this and this PRD is saved in internal documents and now the client requirements is you need to generate.
What else use faces are there? You have product?
Code generation.
And now when it comes to relevance, what you will do is most of the questions,

whatever it is generating, let's suppose the context that you get will be relevant to coding architecture.
So basically if you want these guys to know that you have also built test case generation which is pretty much of the same what product code generation is. So this chunk if you need it then you need diversity right? If you need this context to come in.
So in most of the all relevance, there are high chances this context will be missed out.

Tirth 1:02:39
That is correct.

TJ **Tarun Jain** 1:02:42
So I'm not telling all the chunks that you get are not relevant. So here also if you see this is not that much of deviating, but for the given context, LLM knows even if one context is correct, LLM will look at your question. It will only refer to context one and generate the response.

Tirth 1:02:47
Yes.
Read it.

TJ **Tarun Jain** 1:03:01
But RFP is different. RFP, whatever question you will be, it is not one shot. So what questions are you asking? You know which is single shot.
So single shot is like male contact of Atyantik.
So what is multi short? So multi short basically is you have one question which is X then you have one more question which is Y. So during this time relevance.
Or diversity? Which one do you think is more preference?

Hardip Patel 1:03:33
Yeah, with the.

Tirth 1:03:34
Today, what's your day one?

Tarun Jain 1:03:36

So for multi shot usually diversity is very much preferred and in most of the use cases we'll have multi shot even for single shot. If you have K value to be 5 you are getting diverse as only two.

Which will definitely not harm your available.

Whereas relevance will be 3.

And LLMS are smart enough. If it is single shot question, it will use relevance and generate the output. Diverse is just like a backup thing.

And when you run evals, right, try to run evals for just relevance and try to run evals for relevance plus diversity. I'm pretty sure you'll get more evals for this combination, which is MMR.

And this is for both RFP or any use case you can pick.

Hardip Patel 1:04:36

Yes.

Tarun Jain 1:04:37

Is this clear?

Well, is this done? Oh, everyone got the outputs.

Hardip Patel 1:04:46

Yes.

Tarun Jain 1:04:47

So there are three methods. I hope you guys remember this because you will use this in all the use cases. So I will repeat this again when I do re-ranking. So we still have evals. For evals also we'll repeat the same procedure like how you have to define your query points.

And then we have reranking. For reranking also I will repeat it and then we'll have one separate session for Postgres.

For post base SQL, the logic will be different, but the core concept is same. You index your document. Once you index your document you have to use query. Only the functions will be different, but the functionality is still the same.

Till here is it clear till filter condition and MMR?

Hardip Patel 1:05:33
Yeah.

Tarun Jain 1:05:33
So this is one last part. Like if anyone wants to connect it with Langraf, what you can do is I hope you remember this logic. So from Langraf you have to use state graph then start then end. Now why do I need this type dictionary to define the attributes? So my state is nothing but type predict.
Then context is list of STR.
And STR is there. So this is mainly for attributes. Then we have LLM which is Langchain, Google, Gen. AI, Google generative AI. And after this I just have to define my Google API key.
You have system prompt, then you have user prompt.
And I hope you remember this logic. You have rag state query context answer. I want to manage the state of this. This is where I'm using Langraph.
And your answer generation is same. If you notice you have context, you have LLM, you have prompt, then LLM invoke answer content. But if you look at this logic is different now. Earlier whatever search you used, we used it from.

Tirth 1:06:43
OK.

Tarun Jain 1:06:51
Land chain. Now what is the logic here? The first thing is you're defining your dense and sparse vectors where you define user query. You're defining the user query and then you define prefetch. Once you get the prefetch, what are you supposed to do? That's supposed to define query points. Once you have the query points, you have your relevant documents from payload only get the context and append it in a list.
So now what is your list?
List of STR where this STR is only document.
And then just return the state.
This is the only new thing that we did in the Langra. Based on what we saw earlier, remainings are same.
And where are we starting with? We are starting with the node. I want to add 2

nodes. The first node is search context, second node is answer generation which is search and answer. Then I'm starting with search context which is this keyword. Search context goes to answer generation. Answer generation is my end.
Then graph equals to workflow compile.
Then you have graph and then you just have to do user query. What are the technology and also frameworks used by Athyantic result where you just have to do invoke query, user query. So now if you see it is just two seconds.
Earlier it took 9 seconds and all.
And now you have the response.
I'll share this notebook so that if anyone wants to try the land graph part, you can see this logic of how to define search now.

Hardip Patel 1:08:35
You you already shared it.

TJ **Tarun Jain** 1:08:38
Oh.
This I wrote actually.

Hardip Patel 1:08:42
Yesterday.

TJ **Tarun Jain** 1:08:45
I mean, it's in the vector data. I didn't share this caller before it.

Hardip Patel 1:08:49
Yeah, you're doing this.

TJ **Tarun Jain** 1:08:52
OK, then fine. I thought you created the copy of it.

Hardip Patel 1:08:57
Yeah, we created.

TJ **Tarun Jain** 1:09:02

OK, because I wrote this code yesterday, that's why. But yeah, no worries.
Is this clear how to inference from vector database directly?

Tirth 1:09:14
Yes, we have to update the the line graphic search for using and the diversity. We have to update the search line here for using diversity and MMR.

Hardip Patel 1:09:15
Yes.

Tarun Jain 1:09:21
Oh.
That you can update here. So here. So let's suppose you want to apply filter condition. How will you apply it? Like what logic will you do?

Tirth 1:09:38
With that first query, we would try to figure out what kind of items you have to target.

Tarun Jain 1:09:44
So.
Your voice got broke. Can you repeat again?

Tirth 1:09:49
No, no, please continue. Please continue.

Tarun Jain 1:09:53
No, no, I'm asking like how you guys work.

Ajay Patel 1:09:55
So this is.

Tarun Jain 1:10:00
Hello.

Hardip Patel 1:10:01
I'm here. Yeah, yeah, just a moment.

Tarun Jain 1:10:02
Hello.

RamKrishna Bhatt 1:10:05
Yes, yes, you are.

Tarun Jain 1:10:07
OK, so the problem statement is.

Hardip Patel 1:10:09
It's not.

Tarun Jain 1:10:13
How will you manage?
Filter condition.
Kilangra.
So MMR is very easy. So for MMR what will you do? You will just change these dense vectors to models dot.

Tirth 1:10:35
Nearest query.

Tarun Jain 1:10:39
And then here you can easily add your uh nearest.
As your dense vectors and then you have MMR which is MMR models dot MMR define diverse, define candidates. That is very easy. But what about filter condition?

Tirth 1:10:58
Well, it will start with the. The first thing is you would have to add the keywords and the tags when we are indexing the vectors.

Tarun Jain 1:11:06
Correct.
We need to add value, so where will that value be added? How will we pass this value?
Yes.

Tirth 1:11:19
While we are indexing it, while we are creating or offsetting it.

Tarun Jain 1:11:23
OK, so that value is user input, right? So if it is user input, how will we track that? You understood the problem statement value correct? What approaches is correct? We will define it in filter condition. So for filter condition this value should come from user side or it should be hard coded. So how will you track this?

Hardip Patel 1:11:36
I need to edit other website.

Tarun Jain 1:11:47
Will it come inside node or will it come here?

Hardip Patel 1:11:52
How's that?

Tarun Jain 1:11:57
Oh, you got the point here. So the approach you said was correct, but the only question now is will it come under search or will that value parameter come under state that has to be managed?

Tirth 1:12:10
It will come under state. Actually it won't come with search.

Tarun Jain 1:12:14
Correct. So here what you can do is you can define your very filter and what will be

the data type.
This is correct. Actually it will be dictionary of STRSTR.

Hardip Patel 1:12:26
OK.

Tarun Jain 1:12:29
So what will be your this thing?

Hardip Patel 1:12:34
Or the parent metadata.

Tarun Jain 1:12:35
Dict of field name.

Hardip Patel 1:12:39
Mhm.

Tarun Jain 1:12:40
And then you will have correct field name is nothing but your metadata.
Metadata key and this is your value.
Is this clear? So let's define this. So here what I will do is filter condition.

Hardip Patel 1:12:54
Yeah.

Tarun Jain 1:13:01
Query filter equals to.
Models dot filter.
Then must equals to model dot field condition.
So when I'm defining key, what do I need to do? I need to do.
OK, this is overlapping. So instead of query filter, what I'll do is I'll just add query condition.
So query filter I will get the.
Source.

Hardip Patel 1:13:45
It should be in that key now.

Tarun Jain 1:13:46
OK, wait, how will I get this part?

Hardip Patel 1:13:51
When I ask you, you can do the movie.

Tirth 1:13:51
You can loop through query filter dot keys and then add to the must condition.

Tarun Jain 1:13:57
State. First I need to define state.

Hardip Patel 1:14:01
2nd and I right.

Tarun Jain 1:14:02
State of.
Weight of pill filter.
I need to get the source.
And here I need to get the value.
So whatever key is there, no, this won't work. This will not be proper.
It's better to define source or a defined field name.
As a steer and then value as a steer.
Right, I'll do value of STR.
Because this is the user input source, I don't need it. So what I need to do is I need to extract key from the value.
Alright, how did I do this earlier?
I just define filter condition and then you are doing it OK.
As this also works.
OK, I'll repeat what I was doing it here. So this logic you understood filter condition will come in the rag. I mean your state management optional. I'll copy it here.

And now what you need to do is this filter condition is there it directly pass the filter condition as.
Query filter equals to state of.
Filter condition.
It.
Limit 3 comma.
You understood the logic here. I'm just adding filter condition as a new key and that key itself I'm directly appending it in a query filter.

Hardip Patel 1:16:55
Yeah.

TJ **Tarun Jain** 1:17:00
And now what you can do is answer generation. You can run this, you can run this.
Here when you are defining query here you need to add one more one more field.
Which is your filter condition.
In filter condition what you will do is you will define your entire logic.
Model shot.
Was it filter then must equal to models?
dot.
Feeler condition.
This will be inside square bracket.
And then here you have to define key.
Which is source and match equals the source code.
They didn't give me the name. Why? Because Abortals doesn't have that long knowledge.
Do you guys have that in about us?
Uh.

Tirth 1:18:55
In technologiesauthentic.com/technologies but it is just name of technologies.

TJ **Tarun Jain** 1:19:02
So if I give size development here, what are the all the source I have?

Hardip Patel 1:19:10

That is S development.

Thank you.

Tarun Jain 1:19:16

If I give this, probably you'll get the answer.

I heard it still.

Now you have the answer.

Tirth 1:19:42

Understood. Understood.

Tarun Jain 1:19:46

This will be hard coded but the filter will get inside.

State management because you're passing it from outside. So whatever variables are coming from outside your search, instead of passing it here, which is as a parameter, it's better to add in a state management which can be easily be dragged. So if you want to use this filter condition in another node.

Then you can use that as well.

Tirth 1:20:12

That's good.

Tarun Jain 1:20:14

Cool. So I will let I'm yet to create the template for RFP. I'll do that today, but the approach will be same. So what you guys have to do is you will somehow have to find these documents.

You have questions. You have one document. This will be your input, so when you do inference.

You will have to create a logic of how you will extract the questions from a document and loop over all the questions.

Now for all these questions you will have two inputs. One input is what are the requirements and the second input is your context. So whatever you save inside of Vecta DB will be your this thing. So you have to save both. You have to save your

internal document as well and also the client requirements. So you'll have collection name.
Collection name one will be your client collection name 2.
Will be your internal.
So whatever we have done so far, you can delete those collections so you can open quadrant cloud.
So you can delete all those clusters and you can create these two new clusters. So the clusters that we created if you're using the same document, which is unfortunately not. So that's the reason why you can delete it.
So you can delete both of them.
And internal documents, whatever we did so far, only these four web pages will not be sufficient.
Where is the internal documents?
If you have any other internal documents like PDFs, make sure you have at least 10 PDFs, and if you're using URLs, make sure you add five more URLs. Is this clear? One collection name is for client, one collection is for internal. There is no need to save questions.

Hardip Patel 1:22:23
Yes.

TJ **Tarun Jain** 1:22:23
Questions. If you want to save, you can save it in a metadata.
But again, this is not required for metadata. I'll only prefer summary.
Keywords and the figures.
And then what you need to do is when you look through each and every question, you will first check what is the client requirement and then you will extract the context from the vector DB of internal documents and then LLM will generate the final report.
So I'll create the template for that and I will also add 1 architecture diagram which will be useful for you guys to build that particular POC.
The only new thing is you will create two different collection and then you will inference.
And how you will loop through every question that will be the simple logic like this will be our Python what you call this is like a Python assignment.

Whereas.
How you will use the document internal documents in LLM? It's your rack project, rack person LP.
It's like mostly it's looping. You can also use looping.
Plus, oops, definite.
And feel free to use as long as you know.

Tirth 1:23:44
Can I can I can I work on a different project? I'll give you the definition of the project. Can I work on that as I'm already working on one? Is that fine? It it is going to use the same racket plus NLP. That is what it is going to use.

TJ

Tarun Jain 1:23:51
If in case you guys.
Right, so again for others also, if in case you are building any project which you think is useful, feel free to pick that and if you don't have any project then you can work on this thing. So project if in case you have any of them, you just send me the architecture diagram with some requirements on what you're trying to build. If it is confidential then you can avoid sending me.
You can directly start building it, but just make sure you have drag in it.

Tirth 1:24:24
Understood.

Hardip Patel 1:24:26
Can you share this?

TJ

Tarun Jain 1:24:31
Yeah, this is not required.

Hardip Patel 1:24:33
Good.

TJ

Tarun Jain 1:24:35
This was required. This was not required.

But I'll also send you a template code.
Yeah, you can just refresh this call out notebook. That's it. Anyone has any questions in how to create the filter condition and vector degree?

Hardip Patel 1:25:09
I guess.

Tirth 1:25:16
And like of now.

Tarun Jain 1:25:18
Make sure you use MMR here instead of inspectors, the context tree that we used for the project and for all the use cases because this is this will work.

Tirth 1:25:22
It's.

Hardip Patel 1:25:23
Yeah.

Tarun Jain 1:25:33
Most of the people only prefer for recommendation, but I don't know why, but we usually use it everywhere because we have done evals. So tomorrow we do have the session till 17th, 18th we will start with evals.
So evals and re-ranking are the only two topics that is spending and then Postgres SQL is there. This is like one hour session we'll have just for those who already have Postgres how you can inference it with Blankshield.

Hardip Patel 1:26:04
OK.

Tirth 1:26:10
Thank you. Thank you so much, Tarun.

Tarun Jain 1:26:12
Yeah. Thank you, guys.

RamKrishna Bhatt 1:26:14
Thank you.

Ishan Chavda 1:26:16
OK.

Tirth 1:26:17
Thank you, everyone.

Tirth stopped transcription