Python and AI Power-Up Program Offline Class-20250909_113156-Meeting Recording

September 9, 2025, 6:01AM

1h 36m 22s

Ajay Patel started transcription

Hardip Patel 0:03

I.

TJ

Tarun Jain 0:06

Uh, hello.

Hardip Patel 0:08

Hey, good morning.

TJ

Tarun Jain 0:10

Uh, good morning.

Ishan Chavda 0:10

Type of Google.

TJ

Tarun Jain 0:13

So everyone are here.

Ajay Patel 0:14

Yeah, just a minute. I will be.

Hardip Patel 0:16

Thank you.

TJ

Tarun Jain 0:18

Yeah.

Yes.

Yeah.

Hardip Patel 0:39
This.

Ajay Patel 0:54
Yes, we can start.

Tarun Jain 0:57
Yeah, OK. Uh, so.

Tirth 0:58
Tarun, were you able to go through my WhatsApp message? Were you able to go through my WhatsApp message? OK, OK, OK.

Tarun Jain 1:01
Yeah, yeah, yeah.

Hardip Patel 1:02
Yep.

Tarun Jain 1:04
Yeah, I've checked, so I'll just share this collab.

Hardip Patel 1:12
OK.

Tarun Jain 1:17
It.
So this is the template code and if I come back to this particular pipeline, it's the same thing. What we'll do is we'll start off with document ingestion. So basically what will happen in document ingestion, you have your data.

Hardip Patel 1:27
You.
Mhm.

Tarun Jain 1:36
And after data we have chunking and then embedding and then we have vector database.


Hardip Patel 1:45
Mm.

Tarun Jain 1:46
So we'll follow the same procedure, but here when it comes to embeddings, now we will use both the dents and as well as parts.


Hardip Patel 1:51
OK.
Um.

Tarun Jain 2:05
Is this clear? So the procedure will be same, but here what we'll do is instead of using land chain directly we will use.


Hardip Patel 2:06
Yes.


Ajay Patel 2:06
Yes.

Tarun Jain 2:18
Quadrant or vector is directly.


Hardip Patel 2:26
The.

Tarun Jain 2:26
Instead of using language.
OK, So what we can do is we can use the same data that we want. If not, if you have

any new data and you want to create a new collection, you can feel free to do that. So we'll start off with installation. We need linechain and linechain community only for document loading purpose and once we have the document loading as it.

Hardip Patel 2:53
Yeah.

Tarun Jain 2:54
You can see we are not using blanks and quadrant here.

Hardip Patel 2:55
Yeah.

Tarun Jain 2:58
So what you can do is you can create the copy of this.

Hardip Patel 2:59
So.
Mm.

Tarun Jain 3:20
So inference what we'll do is we'll use Langraph so that we can have the revision again and that inference we will do it on VS code itself. Why? Because the packages are already installed there. So we'll just save the data and then we will use the VS code for.

Hardip Patel 3:21
Stop.
Hmm.
So.

Tarun Jain 3:45
So this process is same again. If I want to load PDFs, what will I do? I'll just directly use line community document loaders, type PDF from 2 loader and if I want to use website I can use web-based loader and if you're using PIE PDF you don't have to do

chunking.
So you can skip the character text splitter part.
And I'll use the same URLs loaders and then chunks and make the chunks.

Hardip Patel 4:11
It.
So.
So.

Tarun Jain 4:25
So here what you can do is this line will be changed if in case you want to use a PDF.
So what will be the syntax?

Hardip Patel 4:26
So.
So.
5 PDF.

Tarun Jain 4:37
No.
By PDFM.

Hardip Patel 4:40
Bad.

Tarun Jain 4:43
Here you just have to add the path of the PDF file.
So if the PDF file is very large and if you are not able to upload, what you can do is you can also mount to your Google Drive. So now what will happen if you click on this? You can upload your PDF files in drive and then copy the drive path here. So if I click on this.

Hardip Patel 4:48
So.

So.
Yes.

Tarun Jain 5:08
Connect to Google Drive.
So whatever data you have, it will be inside my drive.

Hardip Patel 5:21
Hmm.

Tarun Jain 5:26
And you can just copy it from here wherever you have your files. So now I'll just copy this part and I will paste it here.

Hardip Patel 5:29
So.
So.

Tarun Jain 5:35
So now we are directly reading your data from Google Drive by just clicking on this button.

Hardip Patel 5:36
But.

Tarun Jain 5:44
Uh, you guys understood it how I mounted to Google Drive?

Ishan Chavda 5:50
OK.

Hardip Patel 5:50
Yeah.

Tarun Jain 5:51

So since we are only doing offline document processing, typically most of your data will sometimes be in Google Drive. So what you can do is you can upload whatever files you need and then you can directly use it from Google Drive.

Hardip Patel 5:55
OK.

Tarun Jain 6:10
Uh, till here. Is it clear how I got this part?

Hardip Patel 6:12
OK.
Yes.

Tarun Jain 6:14
Cool. So here let's define quadrant client.
From quadrant client input.
Quadrant like and then models.

Hardip Patel 6:35
Mhm.

Tarun Jain 6:38
And what was the another line we were supposed to copy?

Hardip Patel 6:42
From Google Collab import.

Tarun Jain 6:44
So that is also there.

Hardip Patel 6:46
So OK.

Tarun Jain 6:48

Import OS and from.
So if we want to use hybrid search, there was one more parameter here. Does anyone remember?

Hardip Patel 7:03
OK.
Mhm.
Thank you.

Tarun Jain 7:20
So uh, this one vector params is for dense.

Hardip Patel 7:25
Oh, spice.

Tarun Jain 7:28
It's for dense and what is the another keyword? It is parse. So you just have to add sparse and then you will see vector params.
And if I want to use binary quantization, I can directly use it from models which I've already imported.

Hardip Patel 7:41
OK.

Tarun Jain 7:49
These two lines are clear, right?

Hardip Patel 7:53
Yes.

Tarun Jain 7:54
So today we need one more line since we are not using line chain quadrant. So basically how are we using the retriever? So if you look at this code we have retriever then whatever database you have now where is this DB coming from?
It is nothing but your quadrant vector store. So basically what we need to do is we

need to remove the dependency of langchain. So we can't define quadrant vector store because this is coming from langchain and all these things is also from the given library itself.

Hardip Patel 8:13
Um.

Ajay Patel 8:14
Ordering.

Hardip Patel 8:22
Thank you.

Tarun Jain 8:30
So now what we have to do is here. Basically we need to have our own retriever logic, which is our search logic.
So if you want to have the search logic in quadrant, you have something called as points.
And this is very important to note. If you are using langchain, this is the same syntax because they have standardized the process. You just have to define whatever vector store you need, whether it is quadrant vector store, VV8, pine cone or Postgres SQL. You just have to define that store and then you can reuse these functions.

Hardip Patel 9:01
Yeah.

Tarun Jain 9:04
So this is how that library is built, where the functions are same for every single vector store. But here if you're using it separately, VB8 might have their own logic, Pinecone can have their own logic, Quadrant can have their own logic, but at the end of the day the syntax.

Hardip Patel 9:08
OK.

It.
Yeah.

Tarun Jain 9:23
And the terminologies are same. You will have sparse, you will have dense and you will have distance. And what is used in distance?
What algorithm?

Ajay Patel 9:40
M. No, not M.

Hardip Patel 9:42
Yeah, right.
Mm.

Ajay Patel 9:47
Get this.

Hardip Patel 9:47
Compared with this, if I wanna go TFID bottom.

Tarun Jain 9:52
RTF IDF is for sparse vectors.

Hardip Patel 9:57
OK.

Tarun Jain 9:58
Here you can use IDF.

Ishan Chavda 9:58
But it is good.

Hardip Patel 10:02

Uh.
Uh.

Tarun Jain 10:02
Then what was one more?
Does anyone recall two more algorithm?

Hardip Patel 10:07
Yeah.
Uh, cosine.

Ajay Patel 10:11
The same. Hmm.

Tarun Jain 10:12
Hi here it is for sign for distance.

Hardip Patel 10:14
This is.

Tarun Jain 10:18
Beyond 25 and Beyond 2042. Is this clear? These three terminologies. So what we need to do is if you remember the Chroma logic, the first thing is we define the client.

Ajay Patel 10:19
Yeah.

Hardip Patel 10:21
M.

Ajay Patel 10:24
Hmm.
Hmm.

Tarun Jain 10:33

First is define the client.

2nd create collection.

So when you create collection you configure certain parameters, configure parameters, then add documents. So when you add documents now you will have to loop it.

Over the entire chunks.

With the embeddings.

And then search.

So for search we use points. So this is the keyword used in this particular framework.

So if you use Chroma.

Hardip Patel 11:17

Ajayi sutra.

Tarun Jain 11:23

Roma uh, sorry, Roma Gita.

Hardip Patel 11:24

Yes.

Tarun Jain 11:33

So here also it's the same. First you define client, then you create collection. These two keywords will be same. Then you add document, add document logic will be different and for search that directly have a function called query. So some vector database can have collection dot search, some of them have query. So this will change if you're using.

Hardip Patel 11:38

Mhm.

Yeah.

So, so, so.

Tarun Jain 11:52
Using vector database directly.

Hardip Patel 11:54
Right.

Tarun Jain 11:56
And then you have search points and that's it. Once you have search then you can create a node. So node is nothing but your Landgraf node.

Hardip Patel 12:04
OK.

Tarun Jain 12:07
For the retrieval.
So these are the five steps we are supposed to do.
And here for point what I need to do is from client.
Quadrant client dot models I need to import.
Point struck.
So you can just take a note of this.

Hardip Patel 12:32
Yeah.

Tarun Jain 12:36
And since we will be using fast embed directly, what we need to do is from fast embed. There are two things we need to import. One is text embeddings.

Hardip Patel 12:44
Yeah.

Tarun Jain 12:50
So this is for dense and then we have sparse sparse text number.
How did we import this in Langchain? From Langchain we directly used Langchain

community, then embeddings, fast embed and then we use fast embed. But here we need both the embeddings, one is dense and one more is sparse, so we'll directly use it from fast embed.

Hardip Patel 13:09
OK.
OK.

Tarun Jain 13:25
We'll also have a session where I'll show it in Postgres. There the logic is a bit different.
Currently I've uninstalled this in the system, so I have to reinstall it.
Let me know once if it is done or I'll also copy this.

Hardip Patel 13:47
Just a minute.
OK.
Then.
Done.

Tarun Jain 14:13
OK.
So here I've not defined URL and API key, so I'll just define URL equals to user data dot get.

Hardip Patel 14:18
Yes.
Yeah, mm.

Tarun Jain 14:28
Not also name.

Hardip Patel 14:30
Quadrant URL and quadrant API key.

Tarun Jain 14:36
And DPI key equals to user data dot get for DPI key.
And collection name, I'll keep it as hybrid.
Till here is it done?

Hardip Patel 15:14
Yes.

Tarun Jain 15:16
OK, so now the first step is we need to load our embedding model.
Yes.

Hardip Patel 15:19
OK.

Tarun Jain 15:28
So dense embedding model.

Hardip Patel 15:29
OK.
OK.

Tarun Jain 15:38
Text some weddings.

Hardip Patel 15:40
OK.

Tarun Jain 15:41
And then you have model name.
I'll just copy this Gina.

Hardip Patel 15:47
OK.

Tarun Jain 15:57
So this will download those 5 files.
And now same what I'll do. I'll copy this line. I'll paste it instead of dense. I will make it as sparse and for sparse what is the model algorithm? I'll use BM25.

Hardip Patel 16:25
Mhm.

Tarun Jain 16:31
Pars.
Text embeddings. Here you can use quadrant.
VM 25.

Hardip Patel 17:01
Mm.
Done.

Tarun Jain 17:09
Is it done till here?
OK, so here I'll just have to explain one concept. So if I combine embedding and sunks and then save it in a vector DB.

Hardip Patel 17:12
Yeah.

Tarun Jain 17:27
What is this concept part?

Hardip Patel 17:30
Track.

Tarun Jain 17:32
No rag is like you have.

Hardip Patel 17:34
No ingestion, ingestion.

Tarun Jain 17:39
So this process is mainly called indexing or some people also call it as ingestion, right? And here how do we usually calculate the distance measurement? For distance measurement we use cosine.

Hardip Patel 17:44
OK.

Tarun Jain 17:55
So let's suppose I have a query. This query is.

Hardip Patel 17:57
OK.
OK.

Tarun Jain 18:00
Or a movie. OK, and now whatever data you have, it is related to some movie description.

Hardip Patel 18:08
So.

Tarun Jain 18:12
Movie description XYZ all right. You have many movie description. So how did we calculate cosine similarity? We check the vector of query and then cross check with query with document one which is your movie 1.
Mobi.
Document one, then query against movie document two. So this repeats until how many documents we have.

Hardip Patel 18:35

Hello.
OK.

Tarun Jain 18:47
Do you recall this? And after this what did we do? We created a matrix diagram. So in matrix diagram we had diagonals to be 111.
You guys remember this?

Hardip Patel 19:02
Yes.

Tarun Jain 19:03
So we have query. We test the query against every single document to check how similar this is. So this is what you call as pair wise calculation, pair wise distance measurement. But if you look at this approach, this is brute force.

Hardip Patel 19:13
Yes.

Tarun Jain 19:18
Where if your document size increases, you're checking this for every single query and every single document, which is not ideal, right?
So in typical scenarios, whenever you're doing indexing all the vector database.

Hardip Patel 19:31
It.
OK.

Tarun Jain 19:39
Uses one algorithm.

Hardip Patel 19:41
1.

Tarun Jain 19:43
For smart navigation.

Hardip Patel 19:44
OK.

Tarun Jain 19:47
And that algorithm is is called NHNSW which is called IRPL.

Hardip Patel 19:48
OK.
Yes, Sir.

Tarun Jain 19:58
Hierarchical navigatable.

Hardip Patel 19:59
Yeah.
Yeah.

Tarun Jain 20:10
I write here navigable small one, huh?
So now what this will do is, uh, I'll take one example. Let's suppose uh.
What example will you ideal? OK, let's suppose I want to check Varodra.
On a map. So how will you begin with? If you want to check Varadhra on the map, what is your highest level?
OK.

Hardip Patel 20:42
India.

Ajay Patel 20:44
Not India. First we will consider Asia.

Tarun Jain 20:44
This is you start with your country level. Huh. Best you start with Asia then.

Hardip Patel 20:49
M.

Ajay Patel 20:50
Issue.

Hardip Patel 20:51
It's continent level currently.

Ajay Patel 20:52
Then country.

Tarun Jain 20:55
Then country then.

Ajay Patel 20:55
Level.
Then state.

Hardip Patel 20:58
Then state.

Tarun Jain 20:59
Then state.

Ajay Patel 21:03
Then what authorize the district? So yeah, on a level 4, yeah.

Hardip Patel 21:03
And.

Tarun Jain 21:04
Then probably you'll have tweet or anything. Let's suppose I want to check the office itself.

Ajay Patel 21:10
Mm.

Tarun Jain 21:14
So how are you starting with? You're starting with one Asia, which is our highest level, then you're going down, which is country, then state and then St. So this will be considered as level 4, Level 3.

Ajay Patel 21:15
Asia.
Mm.

Tarun Jain 21:29
Level 2 and 11 one. So typically what you're trying to do is you're not checking all the Asia countries.

Ajay Patel 21:32
Hmm.

Hardip Patel 21:32
Yeah.
Got it.

Tarun Jain 21:37
You're not checking every single. I mean, you're not checking all the continents. You're not checking every single country in Asia. You're not even checking every single state in India, but still you will check nearest.

Hardip Patel 21:45
Yeah.

Tarun Jain 21:53
So typically what happens in hierarchical navigatable small world, you have a concept of EANN which will usually check what is the approximate.

Hardip Patel 21:53
But.

Ajay Patel 21:53
Mm.

Hardip Patel 22:01
So.
6.

Tarun Jain 22:05
Approximate nearest neighbors.

Hardip Patel 22:06
Maybe.
OK.

Tarun Jain 22:12
So basically now what will happen is we'll take this movie example. In HNSW you have a parameter called 50. If I define this 50 to be 40, first what it will do, it will check the nearest.

Hardip Patel 22:12
So.
Yeah.

Tarun Jain 22:27
40 movies. Calculate the cosine, which is your distance measurement for those 40 movies. If you found a good match, it will stop. If not, it will get to the nearest.

Hardip Patel 22:32
OK.
Yeah.
Oh.

Tarun Jain 22:43
Check the nearest.
20 movies and then calculate the cosine for those 20 movies.

Hardip Patel 22:52
OK.

Tarun Jain 22:55
OK.
And this is how it will rank and once you have this it will rank. So now how this ranking will happen. If you remember for every single question you have values, you have scores.

Hardip Patel 23:03
It.

Tarun Jain 23:12
So if you see here here you have 111, here you'll have some different value as well, which will be 0.8 or 0.9.

Hardip Patel 23:17
OK.
M.

Tarun Jain 23:21
So what is HNSW trying to do? Instead of doing brute force approach, you're checking the approximate nearest neighbors and then you're ranking the relevant vectors. So you can just call this as smart navigation.
Now why is this important? Every single vector databases uses HNSW by default.

Now the major thing is this M value. So I'll tell you the importance of M because this is very important. Now M value is 16 by default.
Default value. Now if you want to give M value as 30, your performance will be very good. Performance will be good.

Hardip Patel 24:05
Yep.

Tarun Jain 24:11
But it will take too much of memory, but it will consume.

Hardip Patel 24:15
So.

Tarun Jain 24:17
Too much of the memory.

Hardip Patel 24:18
So.

Tarun Jain 24:21
Right if you keep M to be 0.

Hardip Patel 24:25
But.

Tarun Jain 24:26
There is no HRS if you applied.
And you're saving memory.
But performance will not be good. So what do you do in this case? So basically let's suppose you have very large vectors. You will define HNSW configuration to be 0. Once you save the data, you can store the data. Then what you can do is you can update the collection.

Hardip Patel 24:51
Yep.

Tarun Jain 24:55
Update the collection and then make your HNSW configuration.

Hardip Patel 24:58
OK.

Tarun Jain 25:05
Back to M equals to 16. So you guys understood. So if you have very small vectors, that means you have three or four documents itself. You can have M value to be 30, which is more than 16. But if you have very large vectors, what you can do is you can initially keep M to be 0.

Hardip Patel 25:05
Yeah.
OK.
No.

Tarun Jain 25:23
So that you can save the memory. Then what you can do is once the data is saved inside the collection, then you can update the collection and you can keep your HNSW configuration as 60.

Hardip Patel 25:25
It.

Tarun Jain 25:36
Is this clear?

Hardip Patel 25:37
Yeah, so it so that it will then do the HNSW is something like that.

Ajay Patel 25:37
Yep.

Tarun Jain 25:46
So HNSW is mainly you create certain graphs, right? So what will happen when you're saving the data? When you save the data, it is based on document to document, but HNSW is also required on query to document.

Hardip Patel 25:51
Yeah.
M.

Tarun Jain 26:04
So what am I doing here?
It's query against the document.

Hardip Patel 26:11
Mm.

Tarun Jain 26:12
Right, so if your M is 0, you're applying brute force when you're doing the inference as well. Inference in the sense search.

Hardip Patel 26:13
So.
The.
Yeah, got it.

Tarun Jain 26:22
So during that time what you are trying to do when I'm doing inference during that time I want to apply HMSW.

Hardip Patel 26:25

M.
Yeah.

Tarun Jain 26:29
So if you do query.
Cross all the documents which are the nearest documents. Let's suppose I have total 75 chunks.
Out of the 75 chunks I need to get those four chunks right? How do I define that? So it will first look at the smartest navigation like what are the 1st 20 I need to pick which are the nearest. Then I will calculate cosine for that. If in case you got good score then fine. If not you get inside depth. It's like level to level.

Hardip Patel 26:59
Hmm.

Tarun Jain 27:00
First it is the larger level, then you get into minute, so it keeps on getting decreasing level. So the lower the level you are, the more edges you have.

Hardip Patel 27:12
Mm.

Tarun Jain 27:13
Oh.

Hardip Patel 27:14
Yeah, I got it.

Tarun Jain 27:16
Uh, let me also open the document. So every single this thing, right? Uh.
Vector database will have indexing approach so somewhere they should have mentioned HNSW. So if you see for vector index you will have HNSW and the parameters is M where by default it is 16. What is this EF construct number of neighbors considered during the?

Hardip Patel 27:24
OK.
OK.
M.
This.

Tarun Jain 27:42
Index building. OK, so.

Hardip Patel 27:43
So.

Tarun Jain 27:45
Here right when you use this MM is nothing but the depth and how many neighbors you need for that it is EF config.

Hardip Patel 27:47
Yeah.

Tarun Jain 27:54
So this might vary. These values might vary if you're using any other database, but I believe M will be same because M is standard value of HNSW algorithm. So this will be same no matter what vector database you use, but the other parameters will be based on.

Hardip Patel 27:54
Yeah.
But.
Yeah.
Right.

Tarun Jain 28:10
How these folks have applied the logic?

Hardip Patel 28:14
That.

Tarun Jain 28:16
And you can also use this for sparse vectors.

Hardip Patel 28:17
Yeah.

Tarun Jain 28:21
Read the valid code.
OK.

Hardip Patel 28:28
So like for creating collection and updating collection we will use MS0 and when querying then after updating or querying creating we will keep it as 16 right?

Tarun Jain 28:43
Extreme. Yeah. So first here you do create collection.

Hardip Patel 28:44
Yeah, OK.

Tarun Jain 28:48
So everyone have written the students of code.

Hardip Patel 28:52
Yes, yes, yes.

Tarun Jain 28:53
OK, so first now what you'll do is you'll create collection. Here you'll define dense configuration, quantization configuration and sparse configuration. If in case you feel you're taking too much of memory and you need to reduce the memory, what you can do is you can use this HNS config. Can you see this?

Hardip Patel 29:10
2.

Tarun Jain 29:13
HNSW config so you can use this and you can set your M value to be 0. Then what you can do is once you get the green tick mark and you can verify it in the dashboard as well the quadrant cloud. Once the data is saved, what you can do is you can create a new cell.

Hardip Patel 29:14
Yes.

Tarun Jain 29:30
Defined client dot update collection.
Here you can re update your XNSW config.
But you should also define your collection name.
Collection M equals to collection M.

Hardip Patel 29:49
Mhm.

Tarun Jain 29:50
So this has to be done only when you have large vectors.
And not many people usually define this HNSW config.
So create collection.
The first parameter is collection name equals to collection name.

Hardip Patel 30:18
Oh.

Tarun Jain 30:29
Now there are two different configuration. The first one is for dense, the second one is for sparse. So I'll define vectors config.

Hardip Patel 30:35
Um.

Tarun Jain 30:45
Dense.
Then so whatever you mentioned here, make sure you have to reuse the same thing when you define point struct. So if you see here I'm defining point struct. So in point struct you have to define what is your sparse keyword and what is your dense keyword and those keywords are what you are defining in the key.
So now vector configuration I'm defining dense. So there are only two syntaxes. Many people use this as dense. If not, they keep it as the model name. We see Gina AI.
But I usually prefer to keep it dense and sparse so that it's very easy to understand.

Hardip Patel 31:18
OK.
But.
Mhm.

Tarun Jain 31:22
So here you have vector params.

Hardip Patel 31:23
So.

Tarun Jain 31:28
So one is you have size which is 768, then you have distance.
Distance is distance dot cosine.

Hardip Patel 31:39
Oh.

Tarun Jain 31:40
And then I want to run it on this.

I just.
Now what have we defined? We have defined the vector configuration which is dense. These three parameters we have already seen. Now once I close this curly brackets then I have to define.
Sparse. Can you see this keyword? Sparse vectors config. I'll just copy this.

Hardip Patel 32:05
Hmm.

Tarun Jain 32:14
And here what is the keyword first? I'll define sparse.

Hardip Patel 32:18
Yeah.

Tarun Jain 32:20
But.

Hardip Patel 32:21
Mm.

Tarun Jain 32:23
Vector params.

Hardip Patel 32:33
Mm.

Tarun Jain 32:33
OK, so coming back here.

Hardip Patel 32:34
Yeah.

Tarun Jain 32:37
Do we have?

The cosine similarity.
This is BPE.
OK, so let's just recall this. We have step one document document, then you have cosine similarity. So in order to create this thing right one comma one comma one, we used one algorithm if you remember and that was nothing but TFIDF.

Hardip Patel 32:55
So, so.
What?
Hmm.
Mhm.

TJ **Tarun Jain** 33:11
So what basically we'll do is whenever we are defining BM25 or TFIDF, basically in sparse vectors you have to define certain model, right? Model we already know it is quadrant BM25.

Hardip Patel 33:18
It.

TJ **Tarun Jain** 33:28
So the parameters parameters you have to define as IDF. So IDF is nothing but a modifier.
So whenever we are defining sparse vector, you have to define a modifier which is your idea. Even it's there in the documentation. If you look at sparse vector, you have sparse vectors and if you scroll down below you have IDF.

Hardip Patel 33:49
Oh.

TJ **Tarun Jain** 33:58
Modifier which is mainly used for search algorithms.

Hardip Patel 34:02
Mm.

Tarun Jain 34:02
So what we have to do is just copy this line.

Hardip Patel 34:04
Yeah.

Tarun Jain 34:07
Modifier equals to models dot modifier idea. Now if you see the syntax is same here they're only trying to use sparse vectors. So sparse vectors their keyword is text. In our case it is.
Sparse then sparse vector para and paste this particular thing.
So this might be new, but I hope you remember this concept dense and sparse.

Hardip Patel 34:40
Mhm.

Tarun Jain 35:02
Is this done?

Hardip Patel 35:06
Yes.

RamKrishna Bhatt 35:06
Not yet.

Tarun Jain 35:07
OK.
Right.
Is it done?

Hardip Patel 36:13
Yeah, yes.

Tarun Jain 36:13

OK, so now one last thing is that which is our binary quantization. Let me copy it from previous session itself.

Tirth 36:14
I still.

Hardip Patel 36:18
OK.

Tarun Jain 36:22
Binary quantization models then binary quantization. Inside that you have to define a configuration file where always some is false. So just copy this.

Hardip Patel 36:38
Oh.

Tarun Jain 36:42
Binary quantization equals to models dot binary quantization. Binary equals to models. This is correct.

Hardip Patel 36:46
But.
But.

Tarun Jain 36:52
This we have did it in last column which was the rag.

Hardip Patel 36:56
OK.
Mm.

Tarun Jain 37:03
Once this is done, if you run this it should show true.

Hardip Patel 37:10
I don't know.

Tarun Jain 37:47
Is is this done?

Hardip Patel 37:50
Yes.

Tirth 37:50
Yes, it is done.

Tarun Jain 37:51
OK, just one second.

Hardip Patel 37:54
OK.

Ajay Patel 37:56
Yes it is done but on running it is showing me an error client dot create collection. What did I miss?

Tarun Jain 37:56
Right now what we need to do is.

Ajay Patel 38:04
Client collection.

Hardip Patel 38:07
Mhm.

Tarun Jain 38:08
Can you just check for this keywords and uh the comma if in case you're missing the comma or not?

Ajay Patel 38:23
Just.
Just a second.

Tarun Jain 38:26
Yeah.

Ajay Patel 38:27
Stance cosine on this true on this true sparse vectors config.
Because.

Hardip Patel 38:40
Hmm.

Ajay Patel 38:43
Wrong input already exist.

Hardip Patel 38:48
Election. Yeah, you need to change the name.

Tarun Jain 38:49
Oh, can you change the collection name because?
So once you define any collection, right, it will be updated in your quadrant cloud. So either you'll have to delete it and create it again. If not, you'll have to create a new collection.

Hardip Patel 39:03
But.
OK, OK.

Ajay Patel 39:11
Yeah, now it is showing through, yeah.

Hardip Patel 39:12
Hello.

Tarun Jain 39:13
OK, so now if you see in web pages, last time I had 45 chunk. Here in hybrid you have dense which is 768 cosine. Then you also have sparse which is sparse.

Hardip Patel 39:23
Yeah.

Tarun Jain 39:27
And I'll come here and here what I need to do is I need to define points.

Hardip Patel 39:29
Yes.

Tarun Jain 39:34
So points basically is to add your data and once you add your data you just have to use upsert. So upsert is nothing but updating it into.
The collection end. These are the two new keywords. One is points and one more is Upsert. Upsert it is from the client itself. So if you do client dot absert.
This is a function we need to use.
So if you just check the documentation, it updates and creates a new point in the collection. So what we need to do is we need to define this point.

Hardip Patel 40:14
Mhm.

Tarun Jain 40:15
So point equals to.
Empty list and then just you have to loop for IDX comma.
One is you have your dents and beddings.
And then you have sparse embeddings. So whatever autocomplete it has given, that

is correct, but I'll still type it. One is I need dense embeddings, sparse embeddings. How do I save it in a vector DB? So if you remember the diagram from the slides.

Hardip Patel 40:48
But.
I didn't like it.
Um.

Tarun Jain 40:58
So if you have two arrow marks that is going, typically you have to pass our embeddings and chunks simultaneously in your vector database. These two arrow marks CC. So what I'll do is these are the embeddings, both dense embeddings and sparse embeddings. Then I have.

Hardip Patel 41:12
OK.

Tarun Jain 41:15
Chunk.
Which is nothing but my length of chunks.

Hardip Patel 41:22
Hmm.

Tarun Jain 41:23
And chunks of 0 is in document, so I need the page content. So metadata will go into payload and also the content and then the vectorized content will be your page content.
So for IDX dense embeddings, then sparse embeddings sum in enumerate.

Hardip Patel 41:47
It.

Tarun Jain 41:48

Did we cover enumerate before?
This function.

Hardip Patel 41:53
OK.

Tirth 41:54
No, no.

Hardip Patel 41:55
Come on again.

Tarun Jain 41:55
OK, I'll just the syntax. So basically what will happen is.
If you check IDX comma Chang in enumerate.

Hardip Patel 42:06
Yeah.

Tarun Jain 42:09
Songs.
So basically this IDX is there right? IDX will act as a index and then chunk your actual element. So if I just take dummy element.

Hardip Patel 42:15
Hmm.

Tarun Jain 42:25
Maybe.

Hardip Patel 42:28
OK.

Tarun Jain 42:29
And uh.

So now if I print IDX.
It will print from zero to 0123.
What if I print ELE?
It will print many elements. So enumerate is like you have your sequence but it is starting with an index name. So here why do I need index? So basically whenever we are saving it in a vector database you need to have a unique identity.

Hardip Patel 42:55
Mm-hmm.
2.

TJ **Tarun Jain** 43:09
So one is.

Hardip Patel 43:13
She.
Thank you.

TJ **Tarun Jain** 43:14
Let's suppose you have a client. Inside this client you have payload.
And apart from you have vectors.

Hardip Patel 43:22
Mm.
Mm.

TJ **Tarun Jain** 43:25
Vectors is basically your embeddings.
And then you have ID. So ID needs to be unique, right? So basically this was hard coded when you're using langchain and you might have also observed it when you do DB dot add documents.

Hardip Patel 43:34
The.
But other.

Tarun Jain 43:48
And when you pass out chunks, if you observed, what was it returning?

Hardip Patel 43:49
Yep.

Tarun Jain 43:56
I don't know if that output is available or not.

Hardip Patel 43:57
2.
No.

Tarun Jain 44:01
So this line you might have seen some IDs probably you might cross check that. Those are your unique identities. Here what we are trying to do since we have index our ID will be nothing but the IDs which is 0123. Is this clear?

Hardip Patel 44:10
Yep.
Mhm.

Tarun Jain 44:17
So in order to get IDX along with the element, what am I using? Enumerate. So if you check the documentation you have four elements.

Hardip Patel 44:17
Yes.

Tarun Jain 44:31
So can you see this line?
You have zero is a tuple. If I don't do this, if I just print sequence and if I print sequence.

Hardip Patel 44:39
Yes.

Tarun Jain 44:48
It will print 0 Naruto one OP then two DBC. Now what is this? This is a tuple. I know the first index is index. So what will I do? I'll keep it as IDX then comma element.

Hardip Patel 44:56
Hello.
Hmm.
Mhm.

Tarun Jain 45:05
And then near I'll print element.
This is just looping but with index.
Enumerate is looping.

Hardip Patel 45:15
OK.
M.

Tarun Jain 45:18
But with an index.

Hardip Patel 45:19
OK.

Tarun Jain 45:23
Is this clear what enumerator does?

Hardip Patel 45:29
Yes.

Tarun Jain 45:30

Good.
And we need payload, we need vectors which is our embeddings and then we need ID. So for IDX dense sparse and some what I need to do is enumerate.

Hardip Patel 45:35
But.
OK.
But.
OK.

Tarun Jain 45:47
I'll just add.
Zip of.
Why shall I have to convert it to embedding?
We don't have the embedding set, so let me convert it into embedding, then some embedding.
Equals to.

Hardip Patel 46:07
Yeah.

Tarun Jain 46:10
Lens embedding dot embed.
Here what I need to do is for every single document in chunks.

Hardip Patel 46:20
So.

Tarun Jain 46:21
I have to get chunks of page content.
And then I'll just convert this into to list.
And then same goes for sparse.
List of.

Hardip Patel 46:48
Tell.

Tarun Jain 46:48
Sparse embed model dot embed.
For doc and chunks.
You guys know the syntax. What am I trying to do? For every chunk I need to get the page content and I need to convert that into embeddings because here I need the embeddings for every single chunk.
Is this clear?

Hardip Patel 47:23
OK.
Um.

Tarun Jain 47:25
So this will be the embedding suffered.
And if I do the length of dense embeddings.

Hardip Patel 47:31
Transcript.

Tarun Jain 47:32
What will be the size of this?

Hardip Patel 47:35
I'll sort of.

Tarun Jain 47:38
So the length of.

Tirth 47:40
Same as the previous one.

Tarun Jain 47:41

Correct. So the length of chunks is 45.
Then what would be the length of dense embeddings? This will also be 45.
And then I have lentil.
Sparse embeddings.
This will also be 45.
This will take some time, so we are just converting our main content into vectors so that we can add it in the vectors. So for the payload we'll add metadata and the content and then ID.

Ajay Patel 48:22

OK, for here where we have defined chunks.
Can we can you Scroll down?

Tarun Jain 48:26

Tongues is defined here.

Hardip Patel 48:28

Have we created chunks? So if you OK.

Tarun Jain 48:35

So if you see chunks equals to split or split documents, length of chunks is 45. So every single chunk is document and document which has page content and metadata.

Hardip Patel 48:41

Uh, OK.
Um.

Ajay Patel 48:46

OK, actually I uh, it was a dogs over there.

Hardip Patel 48:49

Yes.
OK.

Tarun Jain 48:51
OK, then probably you guys can keep docs if it is in docs.

Hardip Patel 48:53
Yeah, yeah, that's what, OK.

Ajay Patel 48:53
OK.

Tarun Jain 48:55
This will be docs then for doc in docs doc dot page content and then convert it into embeddings.

Ajay Patel 48:59
Mm.

Hardip Patel 48:59
Hmm.
Right. Got it done.

Ajay Patel 49:04
Yeah.

Tarun Jain 49:10
Why is it still taking time?

Hardip Patel 49:21
Swarming also it is low.

Ajay Patel 49:25
Egging time.

Tirth 49:25

It took time, then it crashed.

TJ **Tarun Jain** 49:28

Now it's torn itself.

Tirth 49:29

OK.

Hardip Patel 49:29

I am using people.

TJ **Tarun Jain** 49:32

Oh, what? How long is your chunk size?

Tirth 49:36

Same as yours, Spotify.

TJ **Tarun Jain** 49:39

No, this one it showed 2 minutes. It's done.

Hardip Patel 49:41

What runtime are you using? Could it be that?

TJ **Tarun Jain** 49:43

454545.

I'm not using GPUs, I'm using CPU.

Hardip Patel 49:51

OK, OK. OK, OK.

TJ **Tarun Jain** 49:52

I'm using CPU only.

So if you see the chunk size is same.

Hardip Patel 49:57
OK, one meeting time.

Tarun Jain 50:01
Now what I'll do is I will use a function called zip. So in zip what you're supposed to do is every single function that whatever you have, it should be of same size.

Hardip Patel 50:13
OK.

Ajay Patel 50:19
Yeah.

Tarun Jain 50:20
It is like you are just looping it, but you will get both index and element in a tuple. What zip will do is if you want to run a for loop for three different list at the same time, you just have to ensure it matches the same length.

Hardip Patel 50:20
Oh.
Yeah.

Tarun Jain 50:36
So chunk is 45, dense embedding is 45, sparse is 45. So now what I will I do is I will chunk it in a similar.

Hardip Patel 50:39
But.

Tarun Jain 50:46
Embeddings.

Ajay Patel 50:48
I got an error that sparse embedding model is not defined.

Tarun Jain 50:54
Uh, did you run this two lines?


Ajay Patel 50:56
Which one? Text. Yeah, that one was done. Yeah, yeah. OK. Yeah. Sorry, that one was a typo.

Tarun Jain 51:00
Just check the spelling ones, then some bedding water.
OK, so here is 1 quiz question. If I don't use zip and if I want to use chunks dense embeddings and sparse embeddings directly, then how will I write the logic?


Hardip Patel 51:08
So.

Tarun Jain 51:23
So I can directly do for I in range of length chunks. I know the length of dense embedding chunks and sparse is same. Then what you can do wherever you're using sparse embeddings, you can just use chunks of I.


Hardip Patel 51:26
Um.
Yeah.

Tarun Jain 51:38
Then some beddings of white or some beddings of white.
You can also use this logic.


Ajay Patel 51:46
OK.

Tarun Jain 51:46
Otherwise it's better to use this logic only because we have not seen zip and enumerate before.

Ajay Patel 51:51

This is interesting, huh?

TJ **Tarun Jain** 51:55

Zip is like if you just run this it will create a zip folder. Now you can keep it as K equals to zip.

Hardip Patel 51:56

And.

It.

TJ **Tarun Jain** 52:03

I should do for K in sorry for ELNK.

Hardip Patel 52:06

Oh.

Tirth 52:07

Yes, we also healthy, baby.

TJ **Tarun Jain** 52:10

No.

Ajay Patel 52:11

PHP officer.

Hardip Patel 52:12

Um.

Tirth 52:13

Yes.

Ajay Patel 52:13

But that one is not for file.

Tirth 52:17
No, no, no.

Tarun Jain 52:20
So if you see it will print dense first one it is document because chunk was added first then you have dense.

Tirth 52:21
Uh.

Hardip Patel 52:24
OK.
OK.

Tarun Jain 52:28
And after you have dense if you see not spots.

Tirth 52:30
It is first thing.

Tarun Jain 52:32
And it is in tuple.
So if I do just TL of 0.
I will get the chunk data. If I do one I will have dense.

Hardip Patel 52:43
OK.

Tarun Jain 52:46
If I do too, I'll have sparse.
But we can also go with this logic where we go for the range, length of chunks and then use chunks of I, dense embeddings of I, sparse embeddings of I.

Hardip Patel 53:04
I think PHP does not as it.

Tarun Jain 53:11
What happened?

Hardip Patel 53:12
Collect collection. Sorry, I'm just talking PHP things.

Tarun Jain 53:16
OK so now what I'll do is I will define point make this points.
Now individual point equals to.
Point struct. Inside point struct, what are we supposed to define? One is ID and after ID you have vectors and you have payload. Only three things whatever we have defined here.

Hardip Patel 53:36
Mm.

Tarun Jain 53:38
Payload vectors and ID.

Hardip Patel 53:39
OK.
Mm.

Ajay Patel 53:42
Home.

Tirth 53:44
Right now this is the whole process we are doing for adding the documents in hybrid mode.

Hardip Patel 53:47
OK.

Tarun Jain 53:48
Correct point is for adding.

Hardip Patel 53:50
Yeah.
OK.

Tarun Jain 53:56
I'll just remove unwanted code. This one is clear, right?

Hardip Patel 53:59
Yeah.

Ajay Patel 54:00
Yeah, this one's clear.

Hardip Patel 54:00
Yeah.

Tirth 54:01
Yeah.

Tarun Jain 54:02
Zip I'm not using, but I hope you understood zip and enumerate. Enumerate is like looping with sequence and element. Zip is like if you have three list which is of same length, I will just combine them in a single variable. So now when I use K it will have every single element will have a tuple.

Hardip Patel 54:03
Yep.
Mhm.

OK.
Then.
Mm.

Tarun Jain 54:23
I will also delete this.

Hardip Patel 54:25
Dance.

Tarun Jain 54:26
Now what am I trying to do is now that we have our embeddings in both dense and sparse, I need to add this in a vector DB. So in order to add we have to use point struct so ID equals to IDX.

Hardip Patel 54:32
Right.

Tarun Jain 54:43
Which is incremented. Then what do I have? We have vectors.
So there are two vectors we have and you have to use the same digit you have used earlier. Digit in the sense the text dense.
Dense and then just use.
Then some beddings of I.
Sorry IDs.

Hardip Patel 55:08
Right.

Tarun Jain 55:10
And then what is the second params sparse?

Hardip Patel 55:14
Spars.

Tarun Jain 55:16
Then you have spars and beddings.

Hardip Patel 55:16
Spice 8.

Tarun Jain 55:22
Of IDX and if you remember in sparse embeddings dot zeroth index.
If you see there are two things, one is value and one more is index. So what you need to do is there is a function called as object.

Hardip Patel 55:43
Hmm.

Tarun Jain 55:44
You have to use this Hari.
Is this good?

Tirth 55:51
Hmm.

Tarun Jain 55:52
We need it in this format, so zeroth index we have done. We just have to copy this as index.

Hardip Patel 55:52
Yes.
So.

Tarun Jain 56:00
And paste it here. This is only for parts because parts has two different keys.

Hardip Patel 56:07
Mhm.

Tarun Jain 56:07
Let me know if till here it is done.


Hardip Patel 56:09
So like for uh for as object it shows that it returns none.


Tirth 56:11
It.

Tarun Jain 56:16
What happened?


Hardip Patel 56:17
Like in the documentation of as object it shows that it returns none.
No, no. Yeah, no, I'm using the zip thing.

Tarun Jain 56:32
Yes.
dot 0 dot.


Hardip Patel 56:37
OK, OK. OK. I I think. Yeah. No, no, no. My collab was drunk. Now it's working. Yeah.

Tarun Jain 56:39
No, it's not here.
OK, so is this done ID and vectors and I hope what we are trying to do. I'm looping through every single chunk and I want to use dense and sparse. So dense and sparse is where you have the actual vectors.


Hardip Patel 56:59
Yeah, yes.

Tarun Jain 57:00
Till here is it clear?

Tirth 57:02
Yes.

Tarun Jain 57:03
And here what you can do is now you can define payload.

Ajay Patel 57:04
Hmm.

Tarun Jain 57:08
So payload what I'll do is I will define it as document.
Chunks of 0.
Sorry, chunks of IDX.

Hardip Patel 57:20
Mm.

Tarun Jain 57:24
dot page content and what I'll do is I will define source also.

Hardip Patel 57:24
OK.

Tarun Jain 57:31
Which is chunks of.
IDX dot.
Metadata sums of 0 dot metadata.

Hardip Patel 57:39
Hmm.

Tarun Jain 57:48
Then source.
Huh. I'll just copy this one.

That's it. One is ID vectors payload. I'm using IDX, IDX, IDX, IDX and once you come out of this just append it into points points dot out point.

Hardip Patel 58:00
M.
And.
Mm.
Um.

Tarun Jain 58:11
Now this is the logic to add your data. This will take time. This is where indexing will happen.

Hardip Patel 58:16
Mm.
This looks really clean, the enumerate zip thing.

Tarun Jain 58:24
Extra inputs are permitted X.

Hardip Patel 58:27
Mhm.
We'll find find strip and then.
Oh my God.

Tarun Jain 58:42
What happened?

Hardip Patel 58:45
It works for me.

Tarun Jain 58:46
Input type division.
What are you trying to do?

Ajay Patel 58:56

Yeah, I am also getting the same error.

Hardip Patel 59:00

And what do you say?

Ajay Patel 59:01

Point equal to point struct.

Tarun Jain 59:03

Bens and weddings are 5DX.

Hardip Patel 59:03

Fill required missing.

Tarun Jain 59:10

Is it type?

Oh.

Hardip Patel 59:17

That deck is missing you.

Tarun Jain 59:22

Is this logic correct chunks IDX page content?

Hardip Patel 59:25

But.

Inputs are not permitted. What the heck way?

Tarun Jain 59:34

This is also correct.

Hardip Patel 59:40

There is like a instead of vectors, it should be a vector only.

Tarun Jain 59:40
Then.

Hardip Patel 59:49
Below the I below the IDXID equal to IDX. OK, OK, the 2nd parameter.

Tarun Jain 59:49
What?
Oh, right, right, right. This is vector. This is vector. Yeah, a good point out.

Hardip Patel 59:58
Yeah, OK.
Nice way.

Tarun Jain 1:00:05
And now you just have to define your ad equals to client dot upsert.

Hardip Patel 1:00:07
OK then.
And they accepted the Karan.

Tarun Jain 1:00:16
Points equals to points, then collection name equals to collection name.

Hardip Patel 1:00:18
But.

Tarun Jain 1:00:33
So if I come back here, if I refresh, it is 45211. Yeah, it's added.
Sparse is can you see this length? Sparse is 103. That means it's not fixed whereas dense it is 768. Now here if you see sparse is 98. So do you know what is 98?

Hardip Patel 1:00:43
Yeah.

1:00:45
Mm.

Tarun Jain 1:00:56
What is this 98 and 103? Does anyone recall?

Hardip Patel 1:01:03
Thank you. Um, these are jumps.

Tarun Jain 1:01:09
So if you see this is ID 16.
These are all relevant. So for this particular chunk, what and all IDs are relevant. You can check that here.

Ajay Patel 1:01:24
Mm.

Tarun Jain 1:01:24
So this is where you use HNSW. HNSW is nothing but graphs.

Hardip Patel 1:01:29
OK.

Tarun Jain 1:01:33
So here basically ID if you see 103 that means.
103 nonzero elements remaining all are 0. Here also 98 are nonzero elements, remaining are 0126 are nonzero, remaining are zero same but 768 is same.

Hardip Patel 1:01:43
Yeah.

Ajay Patel 1:01:53
Mm-hmm.

Tarun Jain 1:01:56
So till here everyone are done.

Hardip Patel 1:01:58
OK.

Ajay Patel 1:01:59
Just a second.

Hardip Patel 1:02:00
Yes.

1:02:01
Um.

Tarun Jain 1:02:01
Just to revise what we did, the syntax of this is same. First we start with creating collection. So when you create collection we have to define keywords which is dense and sparse when you're using hybrid search. So what many people do is.

Hardip Patel 1:02:16
Yeah.

Tarun Jain 1:02:17
They also define this as Gina AI and for sparse they'll define it as BM 25. If you do this, when you define your vector here it should be Gina AI and here it should be BM 25.

Hardip Patel 1:02:22
Mhm.

Tarun Jain 1:02:31
Which is also fine and in payload what are you doing? You are saving those data

where you want to filter. This is constant which is your document and then you will also have source. Source is nothing but my metadata. Now what I need to do is.

Hardip Patel 1:02:39
OK.
OK.
But.

Tarun Jain 1:02:46
If there is any question and I want to search only from this particular source file and not from the other source, I can use filter condition. So that's the main purpose of adding this metadata here.

Hardip Patel 1:02:52
I.
So.
Um.

Tarun Jain 1:02:59
And then what was what did we do? We need to save our data inside vector database and this is where you have a concept of point in quadrant vector database which has mainly three things. One is payload and this is not vectors, this is vector.

Hardip Patel 1:03:00
M.
See.
OK.

Tarun Jain 1:03:14
And then you have ID. So in order to have vector you need embeddings. So this is where we converted our entire data which is mainly our page content into embeddings and it is in list format so that we can loop through it.

Hardip Patel 1:03:18

But.
You.

Tarun Jain 1:03:29
And once you look through every single thing, you just have to define your dense embeddings inside vector, which is your Zena AI, basically dense and this is parse.

Hardip Patel 1:03:34
It's.

Tarun Jain 1:03:42
This is dense and sparse.

Tirth 1:03:48
Can you show the enumerate and zip format? The enumerate and zip format?

Tarun Jain 1:03:48
And once you have your appointment, what we are supposed to do?
You want to see that again like for IDX you will have.

Hardip Patel 1:03:59
And.

Tarun Jain 1:04:01
First, I'll keep it as chunk.
Then I have dense embeddings.
Then I have spots embeddings.

Hardip Patel 1:04:11
I I added the code in teams.

Tarun Jain 1:04:13
The name enumerate you will add zip.

Tirth 1:04:18
OK.

Tarun Jain 1:04:18
Of what is my first thing? It is chunk chunks.

Hardip Patel 1:04:21
It.

Tarun Jain 1:04:24
Then it is denser beddings. Then it is sparse embeddings.

Hardip Patel 1:04:29
OK.

Tarun Jain 1:04:31
So if you want to break this down, now what will happen? Your zip is creating a.
45 length element, 45 length element.
Now every single element has three different what you call three different values. So the first one is your chunk, the 2nd is dense, and then you have sparse and you know the logic of enumerate. Enumerate will have zero and sequence, zero and sequence, but this sequence is a zip.

Hardip Patel 1:04:56
OK.

Tarun Jain 1:05:04
Which has three inner elements.
So you understood the logic and then what you can do is point equals to point stuff.

Tirth 1:05:08
Mm.

Ajay Patel 1:05:09

Yeah.
My instruction.
IDX.

Tarun Jain 1:05:16
ID equals to IDX, vector equals to. You'll have dense.

Ajay Patel 1:05:18
It's vector.
Dance colon. Dance M. Dance M. Yeah, first part that's.

Tarun Jain 1:05:25
Just you have to use. There is no indexing here.
And sparse is sparse dot as object. As object is very important because you have indices as well and then you have payload.

Ajay Patel 1:05:34
I'll see him.

Tarun Jain 1:05:46
So payload equals to. You can define text or document, that's up to you. Then you have chunks dot.

Tirth 1:05:46
Mm.

Tarun Jain 1:05:54
Uh, page content.
This chunk is nothing but a page content. Then you also have.
Source which is nothing but chunk dot.
Metadata source whatever the auto complete is and then you can just append this.

Tirth 1:06:16
Understood. Thank you. Yeah.

Ajay Patel 1:06:19
Mhm.

Tarun Jain 1:06:22
OK, so now we save the data. The last thing is query which is search.
And I'll just add a user query mail to contact.

Tirth 1:06:39
Uh.

Tarun Jain 1:06:39
Atyantik and then what will I do? I just have to convert this into vectors. So if you look at the diagram, as soon as you have the query, what is the next step?
You have to convert that into embeddings. We didn't do this in Langchain because Langchain is a wrapper. It takes care of this entire block by itself. So basically you didn't convert into embeddings. What did you do? You directly ask a query and it is going to Vector DB.

Tirth 1:06:52
Mm.

Tarun Jain 1:07:06
In Vector DB, if you see the logic, it has embeddings, so it was converting it automatically inside. But here since we are not using any framework, we have to do this manually. But as per the diagram, so if you see first you have query. After query, what are you doing? You have to convert that into embeddings.

Ajay Patel 1:07:11
Hmm.

Tarun Jain 1:07:25
Then you have to check in vector database which will give you the context. So till here is your retrieval 1-2 and three. So once you have your retrieve context then you can give it to the LLM. Is this clear?

Ajay Patel 1:07:38

Yeah.

Yes.

TJ

Tarun Jain 1:07:40

Oh, I'll just have to create dense embed dense embedding model dot.

Query embed. We just have to give query.

And then you have sparse also since we are doing uh hybrid search.

Sparse vectors equals to sparse.

Embedding model dot query.

Embed.

dot waiting.

And uh, is there any concept of iterator in uh?

PHP or JavaScript?

Hardip Patel 1:08:25

Yes.

TJ

Tarun Jain 1:08:26

So basically if you're using iterators and as you know this is an iterator if you check the data type of this.

Tirth 1:08:27

Yes.

Hardip Patel 1:08:29

Mhm.

Mm.

Tirth 1:08:36

Mhm.

TJ

Tarun Jain 1:08:37

OK.

Tirth 1:08:41

You can just run it in a loop or while.

Tarun Jain 1:08:43

I see this is an iteratable. Can you see this returns? So whenever you are using iteratable, what you're supposed to do is you have to use a function called next only when you are seeing iterator.

Hardip Patel 1:08:45

8 triple.

Tirth 1:08:46

Mhm.

Mhm.

Hardip Patel 1:08:55

Hmm.

Tarun Jain 1:08:58

We saw three new comments today. One is enumerate.

Hardip Patel 1:09:01

OK.

Tarun Jain 1:09:03

One is zip and one more is next. Use only when you have a return object as iterable.

Ajay Patel 1:09:04

Deep.

Iterative.

Hardip Patel 1:09:18

Mhm.

Tarun Jain 1:09:18
And when you are supposed to use it when you have.


Hardip Patel 1:09:19
Yeah.
OK.

Tarun Jain 1:09:23
Multiple lists of same length.
And if you want to loop, you want to loop through all three.
Through through all the list then you use zip. Zip is not required actually if in case you use indexing, but this will like make it readable then enumerate. It's like very simple if you need index along with sequence.


Hardip Patel 1:09:48
M.


Ajay Patel 1:09:49
Um.

Tarun Jain 1:09:54
You need.
Index along with.
Sequence, which is nothing but your elements.


Hardip Patel 1:10:04
Yeah.
Uh huh.

Tarun Jain 1:10:08
And now I'll just run this.


Hardip Patel 1:10:09
OK.

Tarun Jain 1:10:12
And now what we need to do is.
You have to define a logic on how many vectors you need to retrieve, how many vectors you need from dense, how many vectors you need from sparse.

Hardip Patel 1:10:18
Yeah.
OK.

Tarun Jain 1:10:27
Then based on this course, what you're supposed to do is you have to pick top K Top K will be your three. Now how many vectors you need from dense that you can define whether you need it as five or 10 and you're also whether you need 5 or 10. Basically, many people define it as 10, but you get 10 vectors from dense, 10 vectors from sparse, and whichever are the top K match, you will pick top K as three. So this three is your relevant context.

Hardip Patel 1:10:45
Yeah.
OK.

Tarun Jain 1:11:03
Does this make sense? So now what we need to do is we need to define how many vectors you need from dense, how many you need from sparse. And once you have these two things you just have to use search. So when you do search you have to use stop key.

Tirth 1:11:05
Yes.

Hardip Patel 1:11:09
OK.
Hit.
Thank you.

Tarun Jain 1:11:21
So this prefetch.
Equals to.

Hardip Patel 1:11:26
He put.

Tarun Jain 1:11:28
Models dot.
Models dot P Why is it not showing auto complete?
3.
Auto start.
Did I define models?

Hardip Patel 1:11:54
Yeah, yeah, yeah, yeah, yeah. Model dot prefetch collection and vectors.

Tarun Jain 1:11:57
From Cordon client I said model.
I just want to check whether it is small P or capital P.

Hardip Patel 1:12:07
That can be done.

Tarun Jain 1:12:09
Oh, it's prefetch.

Hardip Patel 1:12:12
Mm.

Tarun Jain 1:12:14
So models are prefetch. You have to define your query. So query is basically your dense vectors.
And you should tell using equal to what is the keyword. It is dense and limit is my 10

which is your K value.
Same thing you have to repeat it for.
Uh, spots.
Prefetch.
Very equals to.
Let me print what Ben Specter says.
Dense vectors you need it to be an object. So what you'll do is you'll use models dot sparse vector.
And.
Just pass this vector for your as object.
Now this is using what?
What do I need to add here?

Hardip Patel 1:13:22
Uh, using sparse.

TJ

Tarun Jain 1:13:23
Pass and then limit is 10.
You understood what we're trying to do?

Hardip Patel 1:13:31
Yeah.

TJ

Tarun Jain 1:13:31
Let me check if perhaps slides for that.
So there are just two more lines of code.

Hardip Patel 1:13:49
OK.

TJ

Tarun Jain 1:13:51
OK, so basically now what is happening is when you use a query like neural network, first what you're trying to do is you'll use a keyword search which is your sparse. It will only search for the context which has neural network right? And then you have vector search. Now what vector search will do is whatever is relevant to neural

network, it will also find the.
Nearest for it. This is schematic meaning. Deep learning will also come. AIVs also come. Machine learning will also come. So keyword is basically mainly on frequency of occurrence and vector search is based on the algorithm which is like schematic search. Now what you're trying to do is.

Hardip Patel 1:14:26
No.

TJ **Tarun Jain** 1:14:27
Whenever I enter this query neural network, how many relevant documents you need from keyword search that is our limit which is 10 and how many documents you need from vector search which is also 10 and then whichever makes more sense like which has more score.
What you'll do is you'll combine it.
So models prefetch query dense vectors using dense limit is 10 and you have to repeat the same thing for sparse. Then sparse vectors we have to use as object then using sparse and limit 10.
So this is where your hybrid search logic will be applied.
What we looked into yesterday is this is traditional rag which is very simple and this is one of the first advanced technique where you're utilizing hybrid search.
We can just run this.
Um, understand.
Uh, give double shot. I guess this is keyword search. This is an object.

Hardip Patel 1:15:41
Dogs.

TJ **Tarun Jain** 1:15:43
I hope you guys are on the same line.

Hardip Patel 1:15:47
Yeah.

Tirth 1:15:48
Yeah.

Tarun Jain 1:15:49
So query after query, what are we supposed to do? Convert that into embeddings and once you convert into embeddings, the next goal is to fetch the relevant documents from.

Hardip Patel 1:15:50
So.
OK.

Tarun Jain 1:15:59
Uh, what do you call it?
And the last logic is context equals to.

Hardip Patel 1:16:05
I.

Tarun Jain 1:16:09
Client dot it should be query embeddings.
Claim dot query.
Read points.
Define your collection name.
Collection name equals to collection name.

Hardip Patel 1:16:37
How do I?

Tarun Jain 1:16:38
And then prefetch.
Equals to prefetch.
And here if you want the payload to be searchable, what you have to do is you have to keep with payload to be true.

Hardip Patel 1:16:46
OK, hold on.

Tarun Jain 1:16:55
And with vector. So don't copy this, I'll show you some changes and once I complete this then probably you can copy. But till here I hope you have done query embed vectors then prefetch. Is this 3 lines of code done?

Hardip Patel 1:17:12
Yeah.

Tarun Jain 1:17:13
OK so now the last thing is you have to use query and query points to use search. So basically earlier the command was client dot search but now the command is changed as query points. But let me also search with search.

Hardip Patel 1:17:20
So.
So.

Tarun Jain 1:17:29
It should show deprecated error.

Hardip Patel 1:17:30
So.

Tarun Jain 1:17:34
Don't copy this, I'll show some changes now here with vector. If I do true, it will also print the vectors that it used for search and then limit equals to.

Hardip Patel 1:17:41
What?

Tarun Jain 1:17:48
Top K983. Let me just stop here. Now if I run this.

Hardip Patel 1:17:54
2.
The vectors with vectors S.

Tarun Jain 1:17:58
Oh, I need to define query equation.
Dense vectors.
I'm using them.

Hardip Patel 1:18:12
With vectors the parameter no with payload and with vectors it is just.

Tarun Jain 1:18:27
Oh.
Prefetch is not there.
Using is not there.
We expected.

Hardip Patel 1:18:46
You need to add.

Tarun Jain 1:18:49
So if you see this is a deprecated error. So deprecation warning search method is deprecated and will be removed in future. Use query points. So earlier we used to use search, now we have to use query points.
Collection in prefetch payload any true then query equals to dense vectors.
Which is using.
Dense then with payload is true and if I keep with vector.
With vectors to be true, then top it. Don't copy it, I just want to remove with vectors.
He was from OK.
This is limit.

Hardip Patel 1:19:46
Yeah.
Oh.

Tarun Jain 1:19:52
And now if you do length of.
Context dot points you should get three. Is this clear? Now if I do zeroth index.
It is displaying vectors and I don't need these vectors if you notice because vectors are not required now. It will also show dense and as well as sparse. So what I need to do, I need vectors to be false.

Hardip Patel 1:20:10
Mm.

Tarun Jain 1:20:23
And now there is no vectors and if you notice you have payload which is document and somewhere you should also see source.
I'm very source source.

Hardip Patel 1:20:34
Last line.

Tarun Jain 1:20:35
So now using source what you can do is you can use filter. Since we need to use filter we have to use with payload to be true. If you make this as false it will be empty.
Payload is none, which I don't need, so I also see scores.
Is this clear? Why I need payload? I need payload for filtering.

Hardip Patel 1:21:02
Mm.

Tarun Jain 1:21:04
Payload document. So whatever parameter that you're adding in dense which is in your metadata, if you want that to be searchable, make sure you add that in payload.

Hardip Patel 1:21:07
Yeah.

Tarun Jain 1:21:18
Is this clear?

Tirth 1:21:22
Yes.

Tarun Jain 1:21:23
So this is mainly used for filtering and I'll show how to do filtering in next session and this is the logic for inference. First you have query, then you have dense vectors, sparse vectors, then prefetch. Top case is nothing but your limit and if you want to search the function is query embeddings.

Hardip Patel 1:21:37
Yes.

Tarun Jain 1:21:43
Sorry, query points where you'd have to define your collection name. Prefetch only to be used when you're using hybrid search.

Hardip Patel 1:21:51
OK.

Ajay Patel 1:21:53
Hybrid search means a combination of both sparse and backup.

Tarun Jain 1:21:55
Combination of dense parts in simple words, vectors and keyword vector search. And keyword search.

Hardip Patel 1:22:06
Mm.

Tarun Jain 1:22:08
If you are not doing hybrid search, you don't have to define prefetch and then query is nothing but hey this is the user query. What is what are the nearest documents that you have to return and then using dense then payload is true, then limit is stop key.

Hardip Patel 1:22:09
Yeah.
Yeah.
OK.

Tarun Jain 1:22:26
So now what will you do? This is your context payload which will go inside your node. So now what you can do is you can define search.

Hardip Patel 1:22:38
It's like a little weird.

Tarun Jain 1:22:40
And what is this? This will be state of rack state.
And this is your rack state.
Now what will be your first point?
You just have to copy these two lines, paste it here.
So what will this thing be?

Hardip Patel 1:23:04
Our queries.

Tarun Jain 1:23:04
So this is the line graph. I'm just showing you this index so that you can replace it with the.

Hardip Patel 1:23:07
So.

Tarun Jain 1:23:10
Your existing code base, so this will be.


Hardip Patel 1:23:12
Um.

Tarun Jain 1:23:16
So Ragstad has three variables.
State, sorry context.
Query and answer.


Ajay Patel 1:23:23
Ready.


Tirth 1:23:25
Thank you.

Tarun Jain 1:23:26
So now what is this? This is my state of.
Query. This will be state of query.
Now you have your two vectors. Now what is the next thing? You have to copy this prefetch.
And paste it here.
And then context.
Make this as relevant documents.
And then what you can do is you can define context. So what is the syntax of context? What is the data type of this?
So this is STR. This is STR. But what about context?


Tirth 1:24:12
List of string.

Tarun Jain 1:24:14
Rest of STR. Now what this context will be empty bracket for.

Hardip Patel 1:24:14
Please.

Tarun Jain 1:24:26
For info in.
Relevant documents dot points.

Hardip Patel 1:24:30
Mm.
Mm.

Tarun Jain 1:24:34
Then context dot open info. Inside info you have payload. Inside payload you have your document and then just.

Hardip Patel 1:24:38
Mm.

Tarun Jain 1:24:43
Keep it as your state of context.
Equals to context and then just return state.

Hardip Patel 1:24:51
OK.

Tarun Jain 1:24:52
That's it.
Now this is what you have to use in Langraph and if you notice the speed, this is much faster than Lang chain.

Hardip Patel 1:25:02
Yes.

Tarun Jain 1:25:03

You can also test this out so retriever. It will hardly take 0.2 seconds if you are directly using it from vector database.
And how? What you can do is you can use OPIC, use OPIC and use OPIC for existing code that you wrote with line graph or instead of whatever we rotate, use OPIC first on this particular code base, check the latency.

Hardip Patel 1:25:30
Yep.

Tarun Jain 1:25:31
Then what you have to do is just replace your search function.

Hardip Patel 1:25:33
Yes.

Tarun Jain 1:25:37
Is this clear?

Tirth 1:25:39
Yep.

Hardip Patel 1:25:39
Yes.

Tarun Jain 1:25:40
And here when you're defining embeddings, you have to define both the embeddings.
One is your for text and one more is for.
Spots.
Here if you see there are two embedding model, you have to define both.

Tirth 1:25:57
M.

Tarun Jain 1:25:59

And this we have defined here.
And.

Hardip Patel 1:26:05
Mm.

Tarun Jain 1:26:08
And again, whatever we did here, we'll use the same logic for the existing code bases because this is fast and this is much better way to write and you can use filtering technique as well. So we have two more code base that we need to write in rack and we will use the same logic.
So, yeah, that's it. Do you have any questions?

Hardip Patel 1:26:30
Yes for so last time we discussed like Lang Lang graph was good but that is only for retrieval or or ingestion or as well because like today we did this with.

Ajay Patel 1:26:30
No.

Hardip Patel 1:26:51
Without using langen and it's very very fast I would say.

Tarun Jain 1:26:56
There are two things. Mainly if you want to use vector database directly, I am using this because there is not direct support for the flexibility. So one thing is you can define from vector database, then you can create wrapper around langchain.

Hardip Patel 1:27:02
Hmm.
OK.

Tarun Jain 1:27:14
And then you can reuse the functions like maximum marginal relevance, similarity

search with threshold. But if you know the logic on how to filter by yourself, you don't have to write that, you just have to create a node.

Hardip Patel 1:27:17
OK.

Tarun Jain 1:27:28
Because in Langraph, if you create a node, you can add any logic. It doesn't matter if you're using Langen or Langraph.

Hardip Patel 1:27:29
OK.
Mm.
OK.

Tarun Jain 1:27:36
So that's the reason why in Langraph you can have your own logic. So if you see here, even though this is Langraph, there is no line chain code in this.

Hardip Patel 1:27:43
OK.

Tarun Jain 1:27:46
In this entire function there is no lines in code.

Hardip Patel 1:27:47
OK.
Yeah.

Tarun Jain 1:27:54
So this is one difference that you can have your own logic when you're defining any nodes. And again you can create custom functions, but it's not required.

Hardip Patel 1:27:58
Oh.

Tarun Jain 1:28:04
Because usually you can create a wrapper of the inference. So from here whatever we did right, you can create a wrapper so you can have a custom class and you will have something called as base vector database and then you can do it.

Hardip Patel 1:28:04
Alright.
6.
Mm mm.
Dope.

Tarun Jain 1:28:20
But there is no need to create wrapper. You can have your own logic.

Hardip Patel 1:28:25
Mm.
All right.

Tarun Jain 1:28:27
Oh, any other questions? We'll repeat this. Probably you might have seen something new most of the time, but we'll repeat the same code for at least two or three code bases a second, and during that time probably you'll have most of your doubts clear.

Hardip Patel 1:28:31
M.
Mhm.

Ajay Patel 1:28:41
Mhm.

Tarun Jain 1:28:43
Uh, any other questions?

Hardip Patel 1:28:46
No.

Tarun Jain 1:28:47
You can try to replace this in the existing code base. I've written the logic as well.

Hardip Patel 1:28:51
Yes.
Mm.

Tarun Jain 1:28:54
And if I check the code bases, so I pushed the code in scripts. If you see you have rack streamlet. Yesterday whatever we did right you have it in UTS dot PY. So whatever I do on VS code it will be in scripts.
And whatever we do on Collab, it will be under notebooks.

Hardip Patel 1:29:14
Mm.

Tarun Jain 1:29:14
Under Lancen you have rag uh notebook. I'll also push vector database notebook now.

Hardip Patel 1:29:17
Hmm.

Ajay Patel 1:29:20
Mhm.

Hardip Patel 1:29:21
OK.
Um, there is one more thing. Uh.

Tarun Jain 1:29:29
Oh yeah.

Hardip Patel 1:29:32
OK, so like I was experimenting with this chunking strategy and so like we didn't had overlap when we did it, but when I tried with overlap I had to use like top I use only three and I.
I was able to get good answers, but like if I'm not overlapping then some for some reason I only get good answers like when the top case seven or something like that higher is it?
Uh, like is there any reason you see or uh, could it be specific to PDF just?

Tarun Jain 1:30:13
There we can't make any assumptions because for one or two questions it might work, but what about the bulk ones?

Hardip Patel 1:30:13
What I'm trying to.
Hmm.
Right.
OK.
Mhm.
So.

Tarun Jain 1:30:34
What is agent?

Hardip Patel 1:30:36
Right, OK.

Tarun Jain 1:30:36
So agent will come when we do agent.

Hardip Patel 1:30:41
OK.

Tarun Jain 1:30:43
Schemantic chunking. Uh, this people uses it. Who are these folks anthropic?

Hardip Patel 1:30:49
OK.

Tarun Jain 1:30:50
So basically what you're trying to do is when you're creating chunks, you will also use embeddings and based on their meaning you will use them as a single chunk. This way you're also reducing your length of the chunks. So basically if you notice our code.

Hardip Patel 1:31:00
Sorry.

Tarun Jain 1:31:07
Until we went into vector database, we never used embeddings. We directly used data. Here you'll use embeddings defined schematic chunking.

Hardip Patel 1:31:12
Event.
M.

Tarun Jain 1:31:18
But I didn't see that much of implement in schematic something, but still there are people who uses.

Hardip Patel 1:31:25
OK, um.

Tarun Jain 1:31:25

We have never tried, we just tried it in collab, but once we saw there is no improvements, we never used it in any code.

Hardip Patel 1:31:35
Yeah, just have one question.

Tarun Jain 1:31:37
But yeah, these are experimentations. Sometimes it might work, so mostly I don't play around with chunky much.
Mainly it will be on document loading. If we have proper raw data converted into a text, we don't even have to worry about something.

Hardip Patel 1:31:54
Yeah.
Yeah, I'm just doing it page by page. I don't know if it's alright, but.

Tarun Jain 1:32:01
But yeah, that's fine. Agent tick chunking is very useful because it automatically updates the chunk that you're supposed to pick. So here they're internally using schematic chunking also. I'm not sure, but I'll have to check. But I remember schematic chunking. It's used in many code bases.

Hardip Patel 1:32:10
You.
OK.
OK.
So.

Tarun Jain 1:32:19
The only thing is you need to add embeddings here when you split.

Hardip Patel 1:32:22
Got it. One last question I so I I was checking the quadrant and in in quadrant there is the prompt templates. Is it better to use prompt template what I read?
Is like because it has versioning, it is better. And also one more thing is like quadrant

has this as you showed that it has a way to eval. I don't OK, I saw it somewhere else but it has a way to eval.
Directly from Cortana. Uh, can I do it or it is better to do it myself?

Tarun Jain 1:33:03
Yeah, yeah, you can do. I guess it is MMR. Sorry. MMR is like mean.

Hardip Patel 1:33:06
No.

Tarun Jain 1:33:12
OK, MMR is for maximum marginal relevance. There is something called as MRR. It's mean reciprocal.

Hardip Patel 1:33:17
Mhm.

Tarun Jain 1:33:21
Rank. So this is used to rank your retriever. One is MRR and one more is iterate. You can use it.

Hardip Patel 1:33:26
OK.
OK.

Tarun Jain 1:33:31
It's MRR only. You mean reciprocal rank.
Huh. If you see the second search, it was written DMRR.

Hardip Patel 1:33:38
Great real quality.

Tarun Jain 1:33:42
Uh, it's mean.

Hardip Patel 1:33:44
Mhm.

Tarun Jain 1:33:46
OK.
I mean reciprocal run here. You can use this.
Regarding prompt template, I'm not sure what did you meant by prompt template?

Hardip Patel 1:33:56
Hmm. OK.
So they have also prompt templates. So if if we go under quadrant you can.

Tarun Jain 1:34:06
For which component? Which component?

Hardip Patel 1:34:09
I mean like prompt template as in like they have a way to store prompts there in quadrant itself. Let me just check. Oh sorry, it was Opic. Oh it is in Opic my way.

Tarun Jain 1:34:23
Oh, opaque is different. Opaque you can have your own prompt template. That is for tracing.

Hardip Patel 1:34:28
Mm.

Tarun Jain 1:34:29
There is no prompt template for vector database because vector database doesn't have.

Hardip Patel 1:34:32
Yeah.
My bad, I was talking about Opic.

Tarun Jain 1:34:37
Yeah, there you have from templates that is for tracing. They have their own logic for.

Hardip Patel 1:34:42
Yeah.
So is it like better to use that or not much advantage?

Tarun Jain 1:34:48
No, no, never use defaults. Like if in case people don't want flexibility, they use defaults. But if you think you need your own keywords, some alterations you can do.

Hardip Patel 1:34:59
OK.

Tarun Jain 1:35:00
Even for HNS config I never use 16, we use 2025.

Hardip Patel 1:35:05
OK.

Tarun Jain 1:35:07
Prom template will be there in most of the.

Hardip Patel 1:35:08
I think that.
Then.
Mhm.
And.
Hello.
OK, OK. So I think that runs.

RamKrishna Bhatt 1:35:34
Arun may give us in it.

Hardip Patel 1:35:36
Mm.
But.
Yeah.
Conclude Karuje.

Ajay Patel 1:36:04
I guess Tarun.

Hardip Patel 1:36:06
Message Karinder.

Ajay Patel 1:36:08
Message.
Yeah, he left.

Ajay Patel stopped transcription