And uh as we are starting with Iraq topic today we will have a small intro
0:06
and then move on to the datab databases for geni. Our speaker for today's both
0:11
session is Maxim and we will start with intro as I said Maxim you're very
0:16
welcome to start. Hello uh very happy to see you all
0:21
today. Uh I will start from organizational
0:25
uh topics as well. uh depends on how uh fast we will move. Uh potentially we
0:30
will have some some small break uh between sessions. Uh but let's see.
0:37
Um so uh today mainly we will focus uh exactly on u on the way how uh LLM can
0:48
utilize external information uh for providing uh value uh to the
0:54
users uh that basically utilize uh some systems and we will focus
1:00
uh on the rock uh from the different perspective uh first our focus uh and
1:06
first Our presentation will be uh around general general rock. Why do we need the
1:13
uh rock? Uh and uh some uh more practical details that potentially will
1:20
help you uh to navigate in this uh huge and complex world uh where you can find
1:29
a lot of the tools uh for the different purpose a lot of the databases but you
1:34
don't know where to start and maybe uh where and how to progress.
1:40
So let's uh start maybe a bit from the
1:47
introduction. So I'm a max uh right now at cllum. I'm global head of cloud
1:51
platforms and AI director. So I'm already nine years within cycllum. Uh a
1:58
long time ago, I came here as a middle JavaScript engineer and went through the
2:03
different stages of promotions and I was so lucky to have experience with
2:09
different SDLC uh processes uh and stages um that right
2:16
now I can uh jump much deeper not only in into engineering topics in the DevOps
2:23
topics, cloud topics and in the product topics as well.
2:27
Uh so from the uh start of uh chat GPT era I jumped to the AI direction uh as a
2:36
self-arner uh and I was amazed with the uh some of the experience that I uh got
2:43
uh within the LLM and right now my main interest uh it's to make AI system as
2:51
much reliable uh as it possible that's why maybe you can find uh some of the
2:56
topics uh that I did uh from the previous academies around the prompt

3:02
engineering. This is where I started to uh adopting uh LLM and tried to make
3:08
them more reliable and uh today we will go through uh like
3:14
for for this presentation we will go through uh three uh first uh topics. So
3:20
we will focus on the definition and benefits uh what we have from the rock
3:26
uh some of the main use cases. So from the practical life uh limitations and we
3:32
will uh cover a bit uh what tools you can use uh to speed up this process.
3:39
Uh so in general uh when we are talking about uh AI adaptation landscape so
3:47
basically uh we have our base model and we want to uh direct that base model in
3:57
our for example like domain or our use case or our application.
4:02
uh so we need to start guiding or providing or maybe tuning uh the LLM and
4:11
you always uh go from the simplest way uh of the tune tuning of the LLM to more
4:18
complex um is in some of the like articles you
4:23
can find that even prompt engineering this is some kind of like tuning of the
4:27
LLM because you are changing u potentially like behavior or narrow
4:32
proving the uh direction of the LLM and normally when you trying to adopt LLM
4:39
you are starting from that uh the next stage this is where we will
4:44
focus mainly today uh for both of the sessions uh it's retrie augmented uh
4:51
generation approach uh this is where you already have uh some pipeline uh that
4:59
helps you to add more uh extend and extend uh the knowledge and
5:06
understanding uh of the LLM. Uh the next stage uh it's u like about
5:13
the fine-tuning mainly when you already need to adopt uh model uh in some some
5:21
way or like provide some way of the behavior of the model and for sure you
5:25
can bring more knowledge to the model. And the last one uh the the biggest one
5:30
where you have a combination of rug and basically fine-tuning and rugg uh for us
5:37
uh the main approach when we need to uh to have like some fresh data or our
5:44
proprietary data uh because all of the LLMs have cut off date uh like with the
5:50
tool use uh they can get access to the more fresh uh data from the web uh but
5:57
Sometimes uh it's uh better and easier to provide
6:02

our own data or especially if it's like proprietary and closed data that uh you
6:08
have access only you have access and they are not publicly
6:12
uh and uh it's one of the like most uh effective and cost effective way uh
6:20
basically to uh to let's say fine-tune uh the model
6:26
And uh if we are starting from what exactly is a rock. Uh so this is the
6:33
pattern that uh help us to uh add additional knowledge uh to our model
6:40
like in a in a simple way. We have this like basic rock pipeline uh where we uh
6:47
have like first stage where we edit uh our knowledge and uh vectorize to some
6:54
of the uh vector DB uh and then uh when our user use our system
7:02
together with uh LLM basically
7:07
u first uh request is going to the uh vector database It's doing like some
7:13
similarity search choosing uh few of the uh chunks or like documents from uh that
7:21
vector DB based on how many uh we uh we set up uh and then LLM uh preparing the
7:31
final response to our user. um why it matters uh I I already like
7:41
partially uh explained but in addition uh it's quite uh quite often if you want
7:47
to have like better accuracy in the responses because uh with the usage uh
7:53
of uh of the rock approach you can uh put like exact data uh for the different
8:00
topic that you want to utilize within your system when when user interact with
8:06
that. Uh it's uh quite adaptable because you can uh quite fast change or update
8:16
or add new knowledge uh to uh to that rag database much more cost effective.
8:24
uh and uh it's uh uh in addition it's increase
8:30
efficiency especially uh when you need to provide like big amount of your
8:40
like propriatory let's say data or like knowledge because uh for sure with the
8:46
uh current size of the context window uh quite often uh you can feed uh many huge
8:53
portion of the information uh to the LLM like with all of the knowledge uh that
8:58
you uh want to uh utilize uh as a part of the output than when LLM is working
9:06
on uh that data. Uh but a lot of the research is uh showing that uh bigger
9:14
context windows that you utilized or like bigger prompt uh you you provided
9:20
to the LLM uh you are getting much worse uh result because uh LLM quite often

9:28
missing the exact like needed information in the middle and in
9:33
addition uh you're just providing some um noise together with that data
9:42
like some of the words or information that's uh even not connected to your uh
9:48
basically use case of the database of the communication with LLM.
9:55
Um when we are going to deep dive uh so in
10:00
the rock pipeline in general we have two main stages uh but like when we go a bit
10:07
uh to the details uh quite often and especially this
10:12
information is uh missing when you when you're reading about the uh like rock
10:17
pipeline. It's data preparation for that uh rock pipeline. uh because mainly it's
10:23
focus on like the way of the chunking uh what the database uh you can use and how
10:29
you put the data uh but it's uh quite important to start from the preparation
10:34
of your data uh because I've mentioned uh for example when we provide a huge
10:41
amount of the data uh to the LLM one of the big problem uh that together with
10:48
the data we are providing like water or noise
10:53
uh with uh that's uh just can navigate LM uh when it use uh that like prompt
11:01
information or text uh in the wrong direction. Uh so that's why quite often
11:07
preparation uh of the of your data uh is quite important uh step as well. Uh then
11:14
you have uh the step where you like pre-processing of the data uh like uh
11:24
making like chunking uh process because uh you can you cannot just uh put all
11:30
your 100 page uh documents uh to to the vector database because it will be huge
11:36
and then LLM basically will get uh all of that 100page uh documents uh as
11:44
uh as a prompt from the uh from the vector database. Uh so we we have like
11:49
different approaches uh how to split the information that come with the uh with
11:54
the documents and when we are talking about the documents um the the
12:02
information uh that's coming as a input can be like in many different formats.
12:09
uh because we we we are focusing mainly on the text format but it can be audio,
12:15
it can be video, uh it can be like images uh and different types like even
12:23
uh JavaScript or Python um code bases. Uh then we uh after after the step when
12:32

we uh split our data to the smaller pieces uh we um converting that smaller
12:42
pieces to the numeric value uh and then that numeric value uh basically inest to
12:49
the uh vector database and we have a step of the retrieval
12:55
uh it's when already Our user of our system
13:00
um basically make the request uh and our system based on that request making the
13:08
similarity search from the vector database uh and based on the results of
13:14
that similarity search um it's choosing like top key chunks uh like normally uh
13:23
you are setting that uh it's between like five to 10 top results that you are
13:28
grabbing from that uh vector database and then LLM decide uh what part of the
13:34
information from the chunk it should pick up uh to fulfill the uh the intent
13:40
of the request of the user and provide the output.
13:47
Um so chunking and approach to the chunking is uh quite important part uh
13:54
of this all of the rock uh system uh because uh like
14:03
context window of the LLM LLM it's limited uh for most of the LLMs right
14:10
now for sure we have uh quite a huge one even with within the last Gemini
14:16
versions I mentioning that uh it can uh the context window uh can grows up grow
14:23
up to 2 million uh tokens. Uh but uh still uh the problem with uh missing
14:31
information in the middle uh still uh persist uh with uh huge
14:39
uh with huge uh context that you are providing uh to the model to the models.
14:45
Uh even right now the situation is improving and uh mainly labs are working
14:53
quite heavily uh to improve the retrieval and finding and processing of
14:59
the whole of the information that you provide into the LLM. Uh but still this
15:03
is the problem of reliability so to say and uh within the chunks uh uh you have
15:10
like challenges uh to get the proper balance uh and if you provide like too
15:18
large uh chunks. If you if you split your document uh in a two large pieces
15:25
uh then you will have as a result like not the best retrieval uh or a lot of
15:31
the noise or water uh that you're providing as a part of the request the
15:36
LLM or if you putting it's too small uh then uh you are losing basically the
15:44
main context uh of uh of that information uh because it it can be uh

15:51
splitted in the different parts. It's and it can
15:55
be the case that uh based on the similarity uh search you will not get
16:00
the full context uh of the information and LLM can provide not reliable
16:07
response. Um here just representation of like uh
16:18
three let's say base and main uh chunking strategies. Uh so fixed size,
16:25
semantic, recursive. Uh but for sure you can find uh a lot of different
16:31
approaches in terms of uh chunking. Uh so fixed sized uh it's when we uh split
16:39
um split our information basically in in a fixed size like for example 512
16:46
uh tokens and we uh we having like overlap and overlap uh basically uh help
16:53
us uh to to potentially preserve some context uh from the previous chunk uh
17:01
and it help us uh to increase uh the potential quality of the chunks uh that
17:08
we will have. It is the simp simplest uh approach
17:14
uh and for sure it has each each of the almost each of the approach or I would
17:19
say like each approach uh has uh some uh cons. Um and for for the for this
17:29
approach this is the uh the problem that uh if you always have a standard size uh
17:35
it can be the case uh that your chunk will be break uh in the meat of the
17:40
sentence or uh will not have uh again still the main uh context that you need
17:47
to preserve for this chunk and as a result uh the quality of the retrieval
17:53
stage uh will be much lower um the semantic one here quite often
18:02
some ML system utilized uh and it's trying to split uh all of your
18:08
information by the meaning uh and it has uh much higher like accuracy uh and uh
18:16
in most cases uh much higher uh quality of the retrieval stage because uh we are
18:23
preserving in most cases like context uh that is needed to be in that chunk. Uh
18:29
and when we are doing the similarity search and uh top results LLM is getting
18:36
it's it's getting the proper context for all of the chunk chunks that we picked
18:40
up. Um the uh the problem uh for sure it's
18:46
like slower and more expensive in terms of like uh compute uh and depends on the
18:52
amount of the data that we have. Uh but in general the situation in this
18:58

direction uh with the more compute and uh slower
19:02
um is getting better from the year to year. uh and later on when we will have
19:09
the next session I I will give you uh some perspective on how fast uh things
19:15
uh can can be changed uh with all of that tools that we utilizing right now
19:21
for the agentic system uh and uh in in this uh ecosystem in general uh
19:28
recursive approach it's still uh mainly like coding coding approach
19:33
without the utilization any of ML or AI system. Uh it's uh splitting of the
19:40
information by separators. Uh with this approach uh you basically split we we're
19:47
chunking uh based on the structure of our
19:52
document. Uh so we can identify the paragraphs. Uh we can identify uh the
19:58
boundaries of the document. uh and like for sure it's a bit more complex uh than
20:04
fixed sized and uh still we uh have that uh problem with the context because uh
20:13
paragraphs uh still uh do not represent the full meaning uh of your text or your
20:21
information and this is how uh that three different
20:27
approaches uh looks like. So when we have the fixed size so we have uh two
20:33
close to equal chunks and we have overlap uh of the
20:39
uh of the parts of the sentence uh that we uh have uh quite quite often uh in
20:47
our chunks when we have the recursive. So recursive
20:52
uh just splitted our text uh based on the sentences and uh semantic uh it's
21:01
more focused what the meaning we have uh in each of the part uh of our text that
21:07
we provided. Um right right now uh like for the last time um a lot of the new
21:15
approaches evolve in terms of uh chunking. uh you can uh tr for sure hear
21:22
uh about like aentic chunking uh where we utilizing the LLM and LLM then uh
21:29
provide uh to us more meaningful uh approach uh to this splitting
21:37
uh on the chunks our information um some embedding uh chunks when we
21:44
utilize like exact embedding model and uh based on the uh on the meaning uh of
21:52
the text. It help us uh to uh split the information uh better as well.
21:59
uh but they are coming with uh additional more extensive uh compute
22:05
cost time uh and still uh we don't have like 100%age reliable

22:15
system that uh is the best approach for the splitting chunking or like embedding
22:21
of the information. Uh so you don't need to implement like
22:28
all of that uh that that that we mentioned like approaches to the chunks.
22:32
Uh so some of the top libraries uh that's already uh fully implemented uh
22:40
at least like basic approaches to the chunking. So basically lank chain uh has
22:46
uh has all of uh the basic approaches not only what what I showed uh uh
22:53
earlier in addition uh then they have uh more complex uh chunking uh approaches.
23:01
Uh so it's a quite known uh library like basically application framework uh that
23:10
helping to build the agentic workflows uh and in under the hood it has uh a lot
23:18
of uh additional connectors side libraries uh that's uh helping us uh to
23:25
build that agentic uh system and llama index uh from the start it's mostly was
23:32
focused on the rack specific pipelines. Uh it has quite good performance. It's
23:37
data centric uh and it's just have a bit diff different approach. Uh how they
23:44
implement this uh chankin uh strategy but uh you can uh find even
23:50
uh the same chunk chunking strategy implementation in both libraries. So you
23:56
can uh look there, try them uh and most probably uh you will uh get quite good
24:05
results from uh testing different chunk approaches for exact your uh cases.
24:13
Um in terms of where mainly um rock uh is using uh for sure we can we can talk
24:21
about like from the perspective of general systems and from the perspective
24:25
of the use cases when we are talking about perspective from the use cases. So
24:31
custom support job search detection and you name it. uh where we uh have uh
24:38
quite often updated uh system uh updated uh information with quite often updated
24:45
system and mainly that uh system like proprietary not uh fully publicly
24:50
available especially but when we are talking about exact systems where it's
24:56
mainly utilized it's like for the AI chat bots uh like for example uh if you
25:02
have some support chat bots you are putting uh information as a uh to to the
25:08
rug um like Q&A information or some support
25:14
documents for your clients. Uh search and discovery. So combination of the
25:19

keyword and vector search for better meaning uh of your searches.
25:26
uh different copilots. uh as an example uh it can be even how
25:33
uh many of um current uh systems uh like for
25:39
example when we are talking about like cursor uh or other
25:44
uh they basically use rack for the index indexation of uh the codebase and much
25:50
faster than uh finding the exact places uh in your um in your codebase.
25:58
uh for coding and helping to you. And uh when we need the long context reasoning
26:08
uh it's uh quite often uh utilized uh because again uh the quality uh of LLM
26:19
outputs uh is going down uh with the amount of the data that you feed uh as a
26:25
part of the request. So splitting and finding the exact parts of the
26:29
information uh that you are going to provide is much better approach.
26:36
Um so in terms of like common challenges still chunking and uh context window um
26:44
as we as as I mentioned about the uh problem with the chunking like uh if you
26:51
providing quite large uh chunks uh then then it mean that that's quite large
26:57
chunk then you fit to the model. So we have uh the problem with the uh
27:03
additional not needed information in in exact request. Uh and then we have like
27:11
retrieval quality. So it's based on the uh it based on the exact system uh that
27:19
you that you utilize. I mean like vector database or like approach to the data
27:26
and uh latency and cost. Uh so for sure if we add to any system uh that we are
27:35
developing if we add some additional layer uh way where you need to retrieve
27:40
data or like uh make uh some filtering it's it's coming with the cost of
27:47
latency. uh some dos and don'ts uh for the rock
27:53
system. So uh quite often you always uh quite often uh it's better to start from
28:00
the uh simple uh approach even like from the simple base. Later on we will uh go
28:07
a bit deeper in that uh use the meta data filtering uh in in addition to the
28:14
chunks. uh mo most of the system uh provide possibility to add uh additional
28:20
information and metadata like for example
28:24
uh some category topic uh that's uh then can be the part of your like filtering
28:31
mechanism uh combining the vector and keyword

28:35
search so hybrid approach and about that we will talk uh in our next session
28:42
uh monitor the retrieval quality. Uh in addition to uh to our like ingestion and
28:50
our test uh you we should uh prepare uh some pipeline that that will double
28:56
check the quality uh of our retrieval system
29:01
um evaluate with the domain specific question. So basically evaluation on uh
29:08
that that exact uh system that we are trying to build uh to build like more
29:14
reliable system not rely solely on the vector search. So in addition uh you
29:19
still have uh like your standard searches like keyword and we will
29:25
discuss about uh some of the other approaches that uh providing uh much
29:31
better results uh when they work together uh as a hybrid approach and uh
29:37
rear ranking approach as well. uh one of the biggest problem within the
29:44
rock uh this is the security and access uh control uh so it's quite complex uh
29:51
to implement this uh sometimes but already some of the systems exist and uh
29:58
we can just utilize them overload of the LM context uh so working on the balance
30:05
of your chunks uh it shouldn't be quite big and for sure shouldn't be uh quite
30:10
small because uh based on that you your user user of your system uh will get the
30:17
result that it expected. uh continuous update and uh do not skip
30:23
evaluation framework. Uh and uh here not the three main chunking
30:31
strategies uh but uh the systems uh that's uh or like libraries uh that you
30:37
can utilize like for the scenarios of learning uh the main suggestion is like
30:43
lang chain because you can find a lot of the videos in the YouTube uh it's quite
30:49
extensive community uh around the lang chain and learning uh of the lang chain
30:55
and that possibilities. Uh so basically you can utilize uh chroma potentially
31:01
sent sentence transformers or llama and uh some of the like key reasons uh it's
31:07
uh you can like learn fundamentals and from the perspective or of risk-free
31:14
uh like that stack that uh I provided for the for the learning it's like fully
31:19
open source and you can run just on your laptop but for sure you can uh if you do
31:24
not use any data that's uh uh shouldn't be sent to the uh to the uh models like
31:32

chat GPT and stuff like that you can for sure utilize open AI or like Gemini
31:38
embeddings uh for for the easier approach and like as a MVP uh stage uh
31:45
quite often uh we needed uh just to change potentially like horma uh and add
31:52
uh a bit another embeddings And for the more enterprise systems, um
31:59
it can be the case that L chain is not enough because it's quite good with the
32:04
uh agentic workflows uh but has the problem with some uh agentic
32:10
collaborative uh systems. um and Pineacon and again like OpenAI and
32:17
Gemini embeddings are one of the top uh right now in terms of uh quality
32:26
default configuration. So uh as a start for 80 percentage of
32:33
your cases uh chunk size uh if if you are using like fixed uh chunking
32:39
approach so chunk size 512 uh tokens is enough and golden middle uh for sure
32:46
based on uh your research uh you can identify different size of the chunks
32:53
that working for your uh cases better. uh 15 percentage uh of the chunk overlap
33:01
uh top k retrieval I I would say like 35 sometimes 10 uh and embedding dimensions
33:09
uh between 700 and 1500s uh should be enough uh this data mainly on the
33:18
research of the Nvidia uh and it's kind of like uh golden middle uh middle data
33:25
and where you start when you are building your chunk uh your rock systems
33:31
uh but uh then within the evaluation and testing you can find your exact uh the
33:38
best size for the chunks and for your bra system.
33:45
uh just few uh takeaways uh so rock this is uh the system that helping us uh to
33:55
provide to the LLM external knowledge at the query time. Uh chunking is quite
34:02
critical. So I will not repeat about the exact numbers. Uh hybrid search is quite
34:09
often better than like only vector only search. Uh start always simple not only
34:17
uh for the learning uh when you're starting prototyping all development uh
34:22
you can start from simpler uh systems not uh utilizing from the start the
34:28
enterprisegrade uh databases about what we will talk uh
34:32
later today and uh quite important part it's evaluation of your rock pipeline
34:40
not only like building but system that will help you uh to make the evaluation
34:46
of the quality of your pipeline uh is quite important.

34:51
Uh time for the questions to be honest we already have three
34:57
questions. So first one yeah I believe we have
35:01
already answered it but anyway how do we decide the chunk size for documents?
35:07
Uh how do we decide chunk size for document?
35:11
uh you're starting basically from uh that chunk size uh that I provided uh
35:17
when we have like 512 uh tokens uh and with the sliding window
35:24
uh of uh 15 percentage and this is where you are starting and then based uh on
35:31
your like father experiments quite often when we are talking about like all uh AI
35:38
or like ML development system. Uh you always have part as a research and then
35:45
you uh based on the results that you are getting you are play you are playing
35:50
with the uh with the numbers. So based on what
35:56
you get and what you reviewed or maybe you already built the
36:01
uh evaluation pipeline based on the numbers from the evaluation uh pipeline
36:06
uh you can uh then just change your parameters and see uh if you are getting
36:13
better results and then based on that researches
36:19
uh based on that evaluations uh you can get uh the best number in your case in
36:25
terms of uh what is the size and what is the approach to the chunks should be.
36:31
Thank you Maxim. We have other question from Tani. She said that she find find
36:37
that LLM struggle with analyzing survey data. She had started putting the
36:41
documents as PDF and found that it works better. So the questions are which
36:47
chunking strategy would an LLM use for survey in PDF format versus CSV and
36:54
which do you recommend? Uh we will talk about this uh in our
36:58
next session. Okay.
37:00
So keep that question for the next session in case we will not answer.
37:05
Uh got it. And the next question is rock is just quering additional info from the
37:11
internet and puts it in the context window of your model before providing
37:16
the answer. Is it right? Uh almost except of internet. Uh this is
37:22
our like separate database uh where we are storing our data. Uh so this is like
37:29

the the main pipeline. But for sure we can say that this approach that's uh
37:36
going to the internet and getting additional data this is rugg system as
37:42
well like rack uh system because we augmenting additional information to our
37:48
LLM uh but the base uh when we are talking about the rack system it's
37:54
mainly that we have some uh separate place where we storing
38:01
some information that we want uh our system to utilize as a part of the uh of
38:10
the answer of our LLM. Okay, thank you. And the last one, can I
38:20
put rock to my newly developed GPT and it will be very online and up to date?
38:26
Is it it my newly developed GPT?
38:32
Yep. Uh maybe a person can unmute and just
38:38
say a few words. What does it mean like newly developed GPT? Uh because uh
38:44
yeah so it's it's my question. So I just want to understand do I get it right
38:50
about rugs? So if I have some LLM whatever GPT or whatever
38:55
okay and I want I don't want to retrain it. I
38:59
just want a new information like from the new database. So it means that I can
39:06
add this new shiny layer like rug as I understand it's like chroma or whatever
39:13
and it will query the data from this chroma before
39:17
into the context window and just uh basically
39:21
without retraining I will get all the all the information.
39:26
Yes. Nice. then then then then it's it's
39:30
fully correct uh in in a simple uh understanding
39:36
um maybe maybe I just excuse me may I clarify one point
39:43
yeah sure so so
39:46
I believe the question was asked by Vasili soil used like you're uh I think
39:51
you are oversimplifying it so in the first place you you have to get your
39:55
data into chroma so you have to get your data uh split it into chunks, embed it,
40:03
put it into Chroma and then yes adding uh Chroma and some rag library on top of
40:10
your GPT will work. Uh but you will you will get only your data you will not get
40:16
all the new data available on the internet. For that you would need to

40:20
integrate with some other tools like internet search.
40:23
Yeah. Yeah. Yeah. I I I got it. Yeah. Thank you.
40:29
Okay, thank you so much for your questions. Maybe you have other or we
40:34
can move on to the next part of the session.
40:37
I had one question which is about the beginning part. It's just above my last
40:41
message. Sorry cuz it was at the beginning he mentioned
40:46
that rag injects into the prompt but did he mean the model instead because that
40:50
was confusing me? Uh it's in it in the chat.
40:57
Um okay let's go
41:01
back uh when we have the user query just uh
41:08
to to reiterate uh first step uh in most truck systems where we uh based on that
41:15
query uh our system find out the top uh parts uh of uh our data that we put to
41:24
that uh database. is like five uh potential pieces of the information and
41:31
then together with the prompt and that information
41:36
our uh our system making the request to the LLM and then LLM together with that
41:43
information that we provided from the vector database and the prompted uh like
41:48
user query uh provide the answer uh to the user.
41:55
Okay, got it. So, this what happens on the back end because on the front end I
41:59
wouldn't see that. When I think of prompts as someone who
42:03
is probably nontechnical, I just think of whatever I've entered into the chat
42:07
box like uh for me just like prompt that
42:11
this is anything that we are providing uh to to the LLM all of the information.
42:18
Uh but yes I understand uh like uh just to show as a simple
42:26
example of the rack system as well. So uh when we create uh some projects for
42:34
example in the chat GPT uh the project files that we are adding
42:41
basically that projects then our rack system uh because uh chat GPT just split
42:48
that information to the chunks uh and then utilize that information when we
42:54
are asking uh questions uh in our chat is the simple representation of rack
43:04

system and you can find uh all of that like GPTs uh until you are providing the
43:12
uh expected like behavior or instruction when you just provide the uh the files
43:18
that you want GPS to utilize it will be in addition rack system.
43:24
You know, pretty