LangChain Retrieval Webinar

65 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=VRL7ABRY438
https://www.youtube.com/watch?v=VrL7AbrY438

SUMMARY

The LangChain webinar focused on retrieval methods in the context of language models, featuring experts Omar Khadab from Stanford and Joe Christian Bergum from Vespa. They discussed the evolution of retrieval techniques, the integration of these methods with language models, and the importance of effective retrieval in reducing hallucinations and improving the accuracy of generated responses.

- *Retrieval is essential for connecting data with language models, traditionally using embeddings and vector stores.*
- *The shift from a vector store model to a generic retriever interface in LangChain allows for easier integration of various retrieval methods.*
- *Colbert, a retrieval model developed by Stanford, offers a hybrid approach that combines the efficiency of independent document encodings with fine-grained query-document interactions.*
- *Effective retrieval methods can significantly reduce hallucinations in language model outputs by providing relevant context.*
- *The DSP programming model allows for the creation of specialized retrieval tools that can adapt to specific tasks and improve over time.*
- *Evaluating retrieval systems requires careful consideration of metrics, especially for complex queries that may not fit traditional models.*
- *The integration of metadata and rich document representations can enhance retrieval performance, particularly in smaller datasets.*
- *Ongoing research and development in retrieval techniques continue to evolve, providing new opportunities for improving language model applications.*

AI live welcome everyone to the Lang chain webinar Series today we'll be talking about retrieval a topic that some of us have only started thinking about in the last six months I'd count myself in that category but others of us have been thinking about for years and our speakers today are in that category so we have Omar khadab here from Stanford NLP group who has worked on retrieval models and

Joe Christian bergum here from the Vespa team who's worked on that retrieval engine many many things many features in invest but we'll hear some about that and a lot about the general problem of retrieval and to kick us off we'll Harrison I think you wanted to say a little bit about how retrieval fits with the langjang library what it looks like what people have already seen before we dive into the the the depths that Joe and

Omar have plumbed yeah absolutely I just wanted to and I'll only take like one or two minutes to do this because there's much more exciting things to talk about but just to just to just to ground it in in kind of what what people

familiar with link chain might recognize so I should be sharing my screen now this is this is from a blog post we did in February the the main the main usage of retrieval and Link Chain is basically to

enable connecting your data with language models and so the way that we've traditionally done this is is we've kind of had once we've done everything largely based on kind of like embeddings and vectors and we've taken documents split them into chunks created embeddings for those chunks and then stuff them in a vector store and then we've used that to create a chat bot where you basically have a question that comes in you look up in the vector

store relevant pieces of text you then pass that along with the original question into a language model and it grounds its answer in the documents that it's passed in we discussed we discussed hallucinations last week and and this was one of the the retrieval augmented generation was one of the most popular methods for reducing hallucinations and and a lot of the comments on that webinars basically you know the hallucinations that do still exist come down to messing up on

retrieval and so really excited to learn more about better types of retrieval that we can be doing and and and how we can hopefully you know add some of that in Lang change some of this already is with Vespa and the last thing I'll say here before handing it over is we used to do we used to have a terrible abstraction in link chain which was everything was a vector store and you just did similarity search

in about two months ago we changed that to just being a generic retriever interface which has enabled us to really quickly kind of like integrate Vespa in and other methods as well and so as we talk about some of the ideas today yeah really hopeful that a lot of these can can make them way make their way into an integration with Lang chain in some form and everyone here can play around with it and I'll kind of

stop there because yeah much more interesting people to listen to yeah and I think we had Joe up first to give a bit of a kind of like overview of all the other types of retrieval made possible by not just using a vector store so let's hear from Joe first sure thank you so let me share all right so first of all thank you for having me great being invited here

and see all the excitement of our Lang chain and language models and of course as you said I was in retrieval augmentation for generating with large language models so I'll have a few boring slides I'll talk about information retrieval in the age of genotype AI or large language models and like many of you Builders out there you're probably

building using Lang chain using large language models and you're using a retriever Pipeline and you are retrieving context and you are fitting this into a prompt and you're building stuff and we at the Westport team we're also building stuff using Vespa and using line chain and the reason why we're doing that is that we want to improve our documentation search so here we have several resources thus far related resources around documentation

and our Cloud documentation our sample applications our blog private spa and everything and we thought you know we need to improve the overall search experience so let's connect line chain investment and build a new search interface and in this case also I can answer the question about what is actually information retrieval right so information retrieval is summarized here by a language model based on the routine context on the investment documentation and since we've been working on language models for

search and for search ranking and questioning answering for quite some time so there's a lot of content around that across our sites so the agent does a pretty good job at actually summarizing what information retrieval is and it's the process of retrieving relevant information from a corpus of documents using search and ranking strategies and measuring the effectiveness of these strategies is important right because we want to differentiate and compare different

types of retrieval methods to see you know how they compare and how they perform right so I'll I'll dive into that in the context of doing retrieval for generation we want to find relevant context and we want to stuff that into a prompt or language model prompt and we want to use some kind of information retrieval system right and this might be a search engine it might be a vector database SQL graph hyperdb

numpy ra whatever you know and that's why we love the new abstractions in line train around retrievers and some of the motivation is basically I mean these language models they are large they are expensive and they have limited token lengths so if you cross those multiple different dimensions that's the kind of thing that we want to address by retrieval that might be and of course there's a lot of knowledge in the

parametric ways of the language models and we want to augment that with our private data or data that the model hasn't been really trained on and as Harrison said last last webinar was about reducing hallucinations retrieval improves on that but it doesn't in my experience eliminate it entirely but it does reduce organizations and also allows you to scale to I mean larger data set than just whatever token length limitation your

language model of of references is that so retrieval augmentation is not really a Brand New Concept as far as I know it was introduced actually at back in 2020 at night in Europe's where researchers from meta did build a system where they were retrieve and then they would augment a sequence to sequence language model and generate the answer for open domain question answering and before that just before

that the state of the art and open the main question answering was extractive question answering where you would retrieve with a retriever and then you would extract or predict the best answer Span in the retrieved text so it wouldn't actually generate text it would just pick out the most probable answer from the retrieved context and but obviously now we have much larger models for them I'm just keeping it here we did some work on that we have sample

application to actually reproduce that system and also I think that's the dense facet retriever is just used to Retreat passages from the knowledge base here was using a vector model and I think DPR was one of the first models or papers that described how to efficiently actually train a density removal so yeah and I listened to the

last webinar and there was questions about you know how do you ever make information

retrieval systems and and how do you go about doing that so in Academia there's a set of like common information retrievable data sets that typically researchers will report on and one important collection or is that is the track to retrieval conference which has many collections or relevancy Collections and it's their their effort has been spanning like decades right and for every conference there's a new task

there's deep learning representation there's all kinds of different tasks so that's a rich set of data sets for evolutely relevancy Ms Marco from Bing is the largest or the largest relevancy data set out there with a lot of trading data queries and relevant document pairs there's both a passage version which has like shorter text and there's also a document ranking subtasking where you actually have

web documents and then most importantly I think is the beer Benchmark which is actually a collection of these different information retrieval Collections and the unique thing about the beer Benchmark is that it has all these different types of domains or medical domain web search this different types of tasks and domains and the kind of you go at that Benchmark and you could try a method for instance a ranking model that's been trained on web search like Ms Marco and then you apply

to these new Collections and that basically gives you information about how good is a particular way of doing retrieval when it's applied in a new domain or a new task and then obviously when we emulate these systems we use some kind of metrics and recall that k I won't go into the details on this but basically trying to focus on retrieving all the relevant documents and precision is about really retrieving nothing but relevant among

the top K and Among Us industry practitioners you have the lgdm which is looks good to me right and then we ship it I think that's maybe the most common metric used when we're evaluating informational retrieval systems so in the industry of course there's more than that this we deal with not only static collections like in these data sets we look at engage with metrics if you're running an e-commerce search you will look at conversation sales

revenue obviously not only about relevancy but also multi-objective that you're not optimizing just for the pure relevancy but you also have some other goals a small note on credit distribution is that all these data sets typically have just one query doesn't really weight the queries but in the real production industry application of search and retrieval you will have a very different distribution of queries so you will have head queries that are frequent and

versus the tail queries which are more complex and and you don't see them a lot and I think that the magic of search is in the tale right because that's that's that's really everybody can get the headquers right yo yeah let me ask you a quick question about it though yes that's where you're talking about it so yeah so you started to touch on this a little bit at the end but for for people who are

building applications on their own data I want to evaluate how like different retrieval things are doing on their

own data how would you recommend them to get started like just you know they don't have any data you know they don't they're getting started from scratch right they don't have any data exercises yeah yeah so one thing is to to when you have the BL Benchmark it's it's it's one way to to get an understanding of does any of the beard that the

data sets that are in beer because that closely look like the task you're trying to solve or the data that you have that would be one thing the only obvious thing is that you actually start looking at and and doing queries and then trying to bring out you know people to evaluate the results like are these actually relevant and you are the probably the domain expert so you can you know build a collection of queries and the retrieve results and then

basically annotate is this a relevant or highly relevant and then you quickly can kind of build your own kind of evaluations and that way you can iterate on improving ranking and see if you are improving or not in across these different queries so yeah many organizations do that right and the type of data that they should be labeling there is basically query document and then Boolean is relevant is not relevant for example yeah yeah so in this practical actually

touching the perfect timing I mean here's some practical examples of two relevancy data sets so one is natural questions which is a data set from Google research it has questions that have been asked on Google Search and where the answer can be found on Wikipedia so when did Kendrick Lamar's first album come out and the ground truth is July 2nd 2011. so the way you would evaluate this or transform this question and answering data set into a retrieval problem is that you

will look for the ground suit in the retrieve documents right and if the if there were three documents is in position one okay then you have the recall is perfect because it's you've retrieved the answer in position one and so forth and then you can kind of figure out you know are you actually retrieving the right answer that's actually there and then you can later on I will touch on you know separating the two things about judging the quality of

the generation versus the retrieval quality and another more traditional information that she will data set is track copied which is a data set of covet related literature and one example there is like the how does the coronavirus respond to changes in the weather and in this case you're not looking for a particular answer you could could do that but in this case it's like a classic informational material problem or you have graded relevancy labels and then you could the

task is basically to optimize to retrieve and optimize this ordering and then you can also compute ngcg which is another Precision oriented or you can produce smooth call and I think that famous Marco passage ranking data set and document ranking data sets is the two largest and two most important data sets that are out there because most of the models that you will see even including open Ai embeddings and so forth are actually

trained on this data and it has about nine million passages from from Bing and it has a lot of queries and then you can do retrieval and ranking and then you could evaluate your method and the data set is basically split into three parts so there's a train part and a development part and a test split so you can use those train queries to

train models and then you evaluate your models and

then the models are applied on the same domain and the same kind of distribution that they were traded on which is important and which which leads us to the kind of next about text and retrieval and representations well I mean there's something a lot of focus around dance representations or vector search or vector storage right and searches is is leaning towards dense representations the Legacy or the not legacy but the sparse representation

some people you know and I want to contrast these two kind of techniques that are useful to travel there today and one is the sparse Vector representation where you basically have a very high high dimensional Vector space with millions of Dimensions but for any given text or query or a document there are very few Dimensions that are actually non-zero right then you have and this fits well with a data structure called inverted indexes you can efficiently retrieve over this

and on the other hand we have the dense or the vector and embedding retrieval where we map text into a lover dance vector space and where you can use nearest neighbor search or you can use their nervous indices to speed up and introduce an approximation then I know Umar is Siri will talk about multivactyl representation I'm sure another hybrid combinations and what I think is important when you talk about nectar search and and learning

representations you can also learn sparse representation but I won't cover that but it's it's effectively a way of learning representations right so you want to learn an embedding model that will embed relevant documents for a query so that they are closed in this embedding space right so you're doing representation learning so we have better present language models so they are great we also know more about how to produce good embeddics now but it's

still largely an unsolved problem to have these representations generalized for many different tasks across kind of different domains and there the beer benchmarks this is basically from the beer Benchmark and what the beer Benchmark shows is that many of these dense embedding models that have been trained on the MS Marco labels when they are applied in a different domain when they apply to track covered natural questions and so

forth they actually underperform in terms of ranking metrics compared to like a plain vanilla bm25 that you will get out of elasticsearch or less power a partial solar so so that's one thing that to to to be aware of but embedding models will improve it will be better but I think personally that come behind in these two techniques is is is the future of efficient retrieval because the sparse representations allows you to do exact

matching so you can search for entities or phone numbers phrases and also another thing that's commonly overlooked is that these representations handles variable lengths while dense are Criminal much limited to the input size of your embedded model and plus the dense models are really fast at searching so yeah I have some more on on improving and adoption I'm sure that tomorrow actually is also to kind of talk about

this so and that's the way a new technique that you're actually using language models to improve zero shots ranking so using the language models to actually generate synthetic clear data and I did a blog post on this earlier on and this technique is really an emerging promising interaction and we see a lot of papers and work around this and they've really improve ranking both for dance retrievers or multi-vector representations where you can actually generate training data and

I'm 100 sure that this is the future of how you adapt to your domain and how you make dense models or any types of ranking model better for your domain that you actually use slash language models to generate synthetic data for training and and we do see this also now catching on or having I mean it's been playing out in the industry for some time but Spotify is one example they did a paper on that and I think that's

really interesting too that they add synthetic data to the training their dance retrieval methods and yeah we love Spotify because they use less before for Vector search so yeah so that's an interesting Direction another thing I briefly touched on is that in at scale you need to some kind of have some taste to achieve one ranking to to save costs but that's a separate thing yeah a few practical tips that I've seen I think we need to focus more on the garbage in

garbage it comes out we also need to focus on you know getting the right data into the index and you know how to format it and index it correctly and then you can also do attribution that how you can link the generated answer and like I said earlier evaluate these two Retriever and graduation separately yeah Joe so yeah for the garbage in garbage out and then getting the right data in that seems like very practical and so I'm sure a lot of

people here like what exactly does that mean and what are some like best best practices here right so I think this is standard standard basically from search this is from the thing that we are building and here I'm asking it what are the available pre-trained models that you can use investment cloud and as you can see it doesn't pick up the models because we haven't been able to parse this h correctly right so those type of Errors

you know try to try to avoid that I think one unique experience with this is that we do capture the whole page experience with the links we don't blob this into one snippet and just press enter snippet because for us it's really important that you can attribute the answer and you can click through the sources and you can copy the code or the examples because it it can still hallucinate so I think it's really important that we

actually show a very good representation of the original source yeah so that's pretty much what I what I had so if you edit it you can tweet me on Twitter I'm I'm active on Twitter so yeah I'm really looking forward for almost presentation so yeah thanks thanks Joe I'll be you know canceling you on Twitter later yeah

the question I had was you mentioned these you mentioned doing your like ranking with like synthetic data from language models but I was surprised you didn't mention that earlier than when we were talking about like evaluating your IR system so do you think there's a reason why you shouldn't do that like those kinds of like evaluations or like tests or like things like that with synthetic data and it should only be used for that re-ranking

no I I think you can be used to both but you can ask the the language model to generate I mean there are a lot of options here and we used it in the demo as well to for every passage generate five questions that you think are relevant for this passage and we use that for to complete search and that works really well right so you can actually you know what are the type of questions that people will will ask this

data set you know you can generate that upfront and then if you find able to find these hard negative minings you can also use this to train the ranking model yeah so I think they're they're it's a really emerging topic but the the I'm sure Omar also will present some of of his work on this and it's really promising Direction yeah yeah so yeah you mentioned the hard negative mining which like reminds me of what

you said in the talk about the like tail right the like really difficult queries so I'm curious what are your thoughts on how to ensure like people want metrics they want recall okay and precision at K and not lgtm okay and but like a lot of those are like these statistical metrics that are going to over emphasize bulk of distribution and under emphasize the tale of the distribution these like rare complicated queries do you have any

thoughts about like how to structure your metrics maybe how to structure your continuous integration pipeline or your engineering workflows to ensure that you are not regressing on the tail or improving on that tail yeah I think I think once you get traction once you have actually real user data so that you know what the distribution will be you should basically you know as a startup or you're studying the build on this you should you should focus also you know

getting those head queries you know to stop because that's what has the most impact you know if you cannot get the basic rights people will leave right but so so you see so you should when you're evaluating and that's why I kept that point is that you know you you have to be mindful about these things because in some cases if you're rolling out changes you know you might it might improve the tail but then you are destroying the

head right which will have a virtual effect on on your product so yeah it's it's it's not trivial I would say and that's the one thing I didn't touch on is that these how you're gonna occur how are you planning the using the large larger language models to plan how you're going to do more into the agent plan to planning on how you do in your careers and so on I think that's really interesting and Powerful and and just seeing I I

remember I saw one of the examples from the language chain documentation and I started to a colleague and said you know hey look you know this is super interesting because everybody in search will be talking about query understanding for 20 years but it's generally like hand waving like this but they actually saw a concrete example so yeah and then I wanted to make sure we got at least one of the great questions from the audience we'll we'll talk about them

also after Omar's talk but I wanted to at least prove to people that their questions coming in in the Q a will get answered and there was a really great one from VM which is related to what we've been talking about which is particularly measuring and benchmarking hallucinations so you talked about spans and spans are one way that people have like ensured that there's no hallucination in in like a prior generation of NLP so I'm curious

like is professional techniques for hallucinations do you think that they show up if you just annotate and Benchmark in normal ways do you need special guard rails what do you think foreign I think it's really outside of my core competence to comment on it so I I'll refer to to the actual last webinar on hallucinations but I do think that it's smart to evaluate these separately right so have you retrieved enough

relevant information to answer the question right so you can basically evaluate that and what is the generation that is coming out of that is that hallucinated or is it not right so I think having that separation because that means you might have sold the retrieval problem but then you have more of a prompt problem the language model how you how you tune that yeah so you can separate how you spend the rest of the time improving it

yeah I think that makes sense yeah sort of treating these as separate types of bugs separate like categories exactly exactly yeah exactly okay well then yeah we'll hear again from Joe when we when we have our open q a session at the end but next up we have Omar so let me pull you up here Omar and yeah so Omar was going to share primarily about a specific technique and

some models for this specific technique of multi-factor retrieval so yeah Omar I think you can take it away all right yeah thanks for hosting us Harrison and thanks for joining Charles and you know leading the discussion Joe I really enjoyed your talk you know we've been in touch for a long time since you know you guys integrated Colbert into Vespa one of the very earliest Integrations maybe the very first one so today I want to talk

about effective retrieval models and strategies for knowledge intensive tasks and this you know builds I think very directly on the things that Joe introduced and in particular I want to highlight two things that pertain to what we've been building at Stanford over the past almost four years now so I'll talk about how Colbert which is the Standalone really powerful general purpose retrieval model could fit into you know Lang chain you know chains in the future you know this is

something that could be integrated and could be very powerful there and for the longer tale of sort of more advanced settings how can you adapt a retriever to kind of more nuanced tasks and also react you know to or respond to you know issues that arise and essentially improve the pipelines that you're using within retrieval in say a line chain agent or you know some other chain that you have so this is work with lots of you know folks in The Colbert and DSP

team so this includes you know all the people listed and my advisors at Stanford all right so I'm not going to spend a lot of time on introducing IR again we all kind of know it and Joe did a great job but basically the setting I'll start with here at the beginning is that we're just given this text query and we'd like our retriever to consult some large Text corpus maybe it's our you know private data Maybe it's Wikipedia maybe it's you

know a large chunk of the internet that we've crawled whatever it is and we want it to surface A ranked list of results that maybe we can give later to a language model or it could be part of a more complex pipeline that we're building and you know this is something that we and others have explored in many Downstream tasks maybe you're trying to answer questions maybe you're trying to do fact checking or you're trying to

have a chat bot with a user and you know this is the sort of thing that you know we've been exploring basically during my PhD over the past few years and one of the things that you know working in this space for a while teaches you is that this is a beautiful place for studying balance between quality and efficiency what we want our systems are going to answer challenging natural language you know queries you know in some cases very Advanced

complex questions but they also need to work in seconds and ideally really milliseconds working with millions of documents or you know tens or hundreds of millions of documents as I show and you know when we started working you know you know in in this in this space from a neural standpoint there were two extreme matching paradigms that were sort of plausible and that existed and this is actually kind of like right after you know Bert started being used

in IR so there's this sort of single Vector representation world and this is sort of the Paradigm that Joe emphasized today because this is the mainstream way and most of you here using retrieval methods or maybe almost everybody you know who's not using a Colbert variant is doing something like this in in this in this formulation you have a bunch of documents you're going to run them through a tower architecture maybe birth or something else and you're gonna ask

it to generate a vector maybe a dense Vector maybe a sports vector and then you're gonna index these vectors or store them for search later you get a query in the future you're going to run it through a similar or maybe the same birth architecture or some other hidden proprietary thing maybe open Ai embeddings and it's going to give you the single dense Vector at the end of the day and then you're going to do some

nearest neighbor similarity search this is great in some ways you know I mean we take it for granted but the fact that the query and document are encoded independently means a lot you know you could encode these documents once in advance and cache them and store them and reuse them and that you know dramatically makes you you know able to do things that you wouldn't be able to do otherwise and obviously dramatically reduces costs imagine having to compute

representations for the documents for every new query on the flip side though this is not really you know a very versatile approach the reason being you know you're forced as a model to find this one vector in this one embedding space that is supposed to be able to answer every possible question about a large chunk of text maybe a paragraph or even longer and that's you know if your intuition tells you that's not easy because it's really not easy

it's a very heavy burden on these models and you know this has been the message that we've been sending for for years now and you know it's it's it's it's true as ever so a different Paradigm that sort of have actually you know that predates some of the birth work on you know buying quarters like the ones on the left although not all work on buying quarters is working with cross encoders maybe you've heard of re-rankers I mean recently introduced

some of these so this is an old sort of approach also but differently from the other thing you can see all these dense connections there it pays attention to queries and documents at once extremely powerful fine-grained interactions it it looks at the document in light of the query and so it's very powerful but the problem is it's extremely unscalable you know every time you have a new query all of the documents representations that you've had before are useless because

they've not been conditioned on a new query that you have so this is not scale like if you have millions of documents you know you're not going to want to re-encode all of them for every query that's not practical and this is what motivated Us in late 2019 you know to build cool bear and you know release this actually the same week as DPR you know that you might have heard of where our goal is to keep the best of

both we'd like to have independent encodings we're not going to have like these you know early interactions between queries and documents and so they're going to be encoded independently but key thing is we'd like to have fine-grained representations for them meaning We're not gonna have this bottleneck of reducing each of these into a single Vector because it's really not that plausible you know to to restrict yourself to something like this now the challenge that you face here is

well now you have a matrix you have a set of vectors for that for each document and a set of vectors for each query it's not a single Vector anymore and so standard similarity search is not even well defined anymore what should we do so what we're going to do is we want basically a method that estimates similarity between two matrices but we're not going to do it you know naively what we want is something that

is going to be scalable something that would still allow us to use you know indexes and efficient representations that could scale this to tensor or hundreds of millions of passages in milliseconds so you know what we came up with was this scoring function that is essentially going to take every query term that is encoded as a vector and is going to find the nearest match to that term in that in that document and we're going to

repeat this and we're simply going to sum up these scores and then you know what we're what we're trying to say here is that for every term in the query it's a lot easier to contextually map that into a vector representation because we're just trying to understand one token in its context and that's a much easier task than trying to understand a blob of 500 Words that may or may not you know that you may not know what questions would be

asked about and so with this with this Matrix representation because of the task is is that much simpler these vectors you know there's a whole lot of vectors here but they can be extremely small and they can be indexed for faster so for instance in our early iteration of Colbert you could index all of Wikipedia and search you know and search it in just 70 milliseconds and it's only only been getting faster and you know and and

cheaper to work with so just as an intuition building exercise here here's a real exercise of matching say you wanted to answer the question when the Transformers cartoons just come out you know not the the attention is all you need paper so this is the query and I have Snippets from the document here the relevant document and if

you look at what Colbert does at matching time it encodes the word when with a vector representation that's very close to the

on in you know in in the answer that talks about the date you know certainly it gives Transformers as you know a similar representation in both the query and the document cartoon matches with the verb animated very reasonable you know and release matches with come so you know does this actually work and what we learned back in you know late 2019 early 2020 is that you know compared with concurrent approaches that use Bert also for you know single Vector

representations things that are really popular DPR or NC Colbert you know is a whole lot better in Downstream matching in fact it preserves all of the accuracy of the re-rankers that are really expensive you know that I talked about while being you know obviously orders of magnitudes faster than cheaper and you know this is something that you know you reliably see on on many tasks the interesting thing though that we didn't think of when we built Colbert

was that when there's a domain shift when you're which is something that Joe highlighted when you're trying to move from the training domain say Ms Marco or natural questions and you're trying to test these models in new settings the interesting thing that I think was first highlighted by beer in their evaluations was that you know these interaction models in particular late interaction or you know the expensive interaction they're a lot better at sort of transfer from one domain to the other and our

intuition for that is encoding context at the level of tokens is just a whole lot easier and more robust and resilient than you know you know cramming everything into one vector no matter how large you make it and you know ever since there's been lots and lots of studies sort of confirming like interaction is generally a lot better than you know single Vector retrievals for search but we've also been doing work and others have been doing work applying this to

Downstream tasks so this was called bear V1 you know if you're gonna use this we strongly recommend Colbert V2 which we built you know in late 2021 early 2022 and we focused here explicitly on trying to get models that generalize better V1 generalized better to new domains just by happenstance it works better but could bear V2 was purpose Built For This by you know a more resilient denoise training strategy look you know I I refer to the I refer you to the

paper for the details of that but we also built this residual compression technique so that each of the vectors in The Colbert storage are really small so if you work with Colbert B2 right now even though we have you know more than 10 times as many vectors as you're you know like off the shelf by encoder the vectors are actually taking no more space than a standard you know like VPR retriever or you know this single

guy to retrievers because each of our vectors could be included in as little as 20 bytes or something like that so you know we have applied this in domain and out of domain so the the two tables and out of the main deaths a whole bunch of them beer we also introduced this latte you know a a play on the words there and you know in

that cross you know I think it was something like you could

count them but it was like 22 out of 30 evaluations you know it it comes out on top across a wide range of you know sparse and dense retrieval models that are you know very recent and Powerful so you know the the the the the Colbert V2 introduction which is you know what's what's what's on the you know Colbert Report right now you know it reduces the index size quite a bit over Cooper V1 and obviously maintains

the same highly efficient you know latency and Survey more recently than that we introduced dedicated retrieval engine like because we have this different interaction mechanism like interaction and what we wanted is you know to build an engine that can support efficient serving of these models at Large Scale or efficient search at Large Scale so this is something that you know we spent a bunch of exploration on and you know just to to highlight a few of

the you know of the results there so if you're working with something like Ms Marco this is probably larger than most of the you know local data sets you might work with so it has half a billion tokens with a B 9 million passages you could encode this you know in in in colder V2 was pled and as little as 25 gigabytes and you can search even if you don't have a GPU in just 45 milliseconds for each query

Wikipedia a bit larger but you know the entire index could be around 100 gigabytes and you could search you know with a CPU and as little as 67 milliseconds and if you have a GPU it only gets faster and then the largest data set we tried was Ms Marco V2 which has close to 10 billion tokens 140 million passages and the index is still you know just 200 gigabit gigabytes and you know you could still search this on

a CPU you know in just a little over 180 seconds and obviously with the GPU again a lot faster so this is you know public you could just go to the Colbert repository and play with this this is something that maybe you know will be exciting to chat about you know Integrations of something like this into into line chain I guess you could actually already use versions of this through Vespa in line chain so that might be an interesting you know a

different way to to to to to to do to do a lot of this I was going to ask exactly that like what is the what is the easiest way for people to to play around with this and like what would that involve so if you know if I go to that GitHub repo do I have to like host the the kind do I have to like run the embedding model like locally myself and host that and it

sounds like maybe there's something in Vespa that takes care of that are there other like what's the is yeah what's this what's the simple or what are the what are the options for people to use this and what would those entail so cool there is is more of a you know it's just a you know it's a model so there's a checkpoint and then there is the encoders and then there is the infrastructure that we have for serving

the infrastructure here being software infrastructure if you want to host Justice or or such either you dedicate

your own machine with CPUs for search and gpus for indexing or you know Vespa might be you know your best shot otherwise add this in terms of a hosted a hosted I'm not aware of any other hosted versions of this but you know certainly you know I I hope that looking at various results that I have she I've shown and will show up more of

will encourage more people to sort of build infrastructure around this the challenge being you know you get a lot of quality and it's fast and it's efficient but it's a different infrastructure so that's the that's where something like vespa's flexibility I think the query language you know didn't get a chance to highlight that but it's maybe not as much as as we'd like but you know it's a very powerful query language that makes that possible but actually I I so I'm happy to take

more questions about that but I wanted to highlight sort of a kind of a bigger picture story here so suppose that you you know you're not working in isolation and I think a lot of the questions we're talking about scenario videos where you just don't have a lot of data and you know you can't just like train you know your own retrievers and such and you know you could just use Colbert out of the box zero shot but could you do

better and so we spent the last two to three years looking at like how we can apply Colbert to new domain so I mentioned Kobe V2 already for zero search but if you wanted to answer questions with language models you know we've looked at Colbert Q8 which is this sort of pipeline for essentially very similar to rag but has a different supervision technique and uses Colbert and that that delivers large gains against things like that we also built this system hindsight

led by Ashwin here at Stanford which focuses on trying to build chat Bots with colbert-like models and how do you make it grounded and check for hallucination and things like that so people who are looking for ways to evaluate that definitely check out hindsight and we looked at also just fact checking especially in multi-hop and complex questions and you know I don't want you to go and read each of these papers now unless it's exactly relevant to you but what based

on but what I want to say is based on these sort of building these different systems on top of Colbert or colbert-like models we've been working for the last year or a little bit more on what you should use in 2023 to adapt retrieval maybe it was Colbert or with something else to this to these fast changing data scores fat you know sort of constantly you know evolving Downstream tasks and for that you know you could be working in line chain and

you have all kinds of you know agents and tools at your disposal an additional one that you should consider is the DSP programming model and for land chain audience you could think of the SP as this Advanced tool builder for highly specialized retrievers so you know you're adding tools all the time to to line chain most of them work out of the box but let's say you have this really difficult task where you want to specialize retrieval to work with you

know specific requirements on length or dependencies in the sense of multi-hop queries or you know you see a

lot of issues that users report and you want to tackle them working with prompts or chains or you know other abstractions can only take you so far at least in a systematic way and this is what we build the DSP programming model for so this is something that you know you could build tools with that you know are used by

your land chain agent and at a high level you know I can't really think dig into all the details here but at the high level you're going to write a small program which is a python function in you know just plain old python with standard control flow to describe what you know what you'd what you see your Retriever as doing in interactions with a language model and I'll I'll sort of touch on some examples of that and the

key thing is in that in that program that you're writing which might have loops or you know if statements or just you know various complex control flow elements you're not going to write prompts and you're certainly not going to write few shot you know demonstrations for your prompts instead you're going to just focus on describing the high level Pipeline and the leaves of your pipe line are going to be these declarative calls either to a retrieval model or to

a language model and you're going to use built-in Primitives in DSP that will take your program and a language model and one or two you know and a small number of like end-to-end task labels that you can hand annotate and I'll show examples of that and it will build up few short pumps for your task or it could do even you know more interesting things than that like for instance the same program that you write without

changes can work as a zero shot sort of like pipeline it can work as a few shot pipeline without hand annotating few short examples but it can actually compile itself into a small language model that you run locally saves you on cost makes makes you able to adapt things a little bit more sort of concretely to your to your to your needs and you know we we have a paper on this so feel free to check it out on the

SP and you know we've tested this out on a bunch of tasks with you know I think amazing results so you know for instance what what that highlight now is if you're working on Hotspot QA so let's say that you're building you know a conversational agent you know and maybe it can access many tools maybe calculators and other things but one of the core functionality is you want that agent to be able to answer complex questions on your data

whether Retriever and maybe these complex questions you know maybe just using an off-the-shelf retriever even if it's Colbert is only going to take you so far because there are inherent dependencies there are questions that you can only answer by answering simpler decompositional questions and you know this is something that a lot of the you know react agents and whatnot in in Langston can help you get started on but what they can't really get you to do is

how to teach them to use an arbitrary tool like you know react does not know how to use kuber B2 so you're kind of forced to use a zero shot react agent or maybe to sit down and try to write prompts and hope they work there's also challenges of like okay if you want to use like react and make it reliable maybe you gotta use like gpt4 or at least an expensive you know GPT 3.5 thing in production and maybe

you don't want to you know you know put that all out there and you know not have a lot of ways of controlling it and you know what we what what what DSP gives you are these magical Primitives that would take your program which is describing this retriever which could be a tool of your line chain agent and you basically just say hey dspi it just implemented this declarative pipeline that is maybe a zero shot DSP based react agent

just like a copy of the version that's in line chain but now I actually wrote two examples by hand of some of the questions that users ask and I have defined that answer I definitely don't want to spend time like writing prompts or like intermediate pipeline specific data but I just you know I have these examples and you can treat them as unit tests essentially and make sure that whatever pipeline you have can pass those reliably so you could just say

dsp.d demonstrate and give it the function program that you wrote in in DSP and give it these two labeled you know end exam and task examples and what we've you know what we what this would do for any orbitrary DSP pipeline that you give it you know and you can sort of customize the way demonstrations are done is you know it can it can teach the language model how to interact with the different pieces of the pipeline

that you've built so for instance you know this this sort of like demonstrate line that we have here can convert the zero shot react agent that's written in DSP in a few lines of code into a few short react agent that knows how to interact with Colbert V2 and in practice this can raise accuracy on you know benchmarks like hotspot QA from around 30 in a zero shot setting to a 35 in a few shot setting

that you did not write the prompts for so it can you know teach itself how to generate queries and how to use the passages and you know how to control the flow Etc and you know you could you could easily add you know extensions to the pipeline that keep raising this you know a very simple like a few line addition makes this 41 exact match so another sort of magical features and I'll conclude with this is the compile primitive in DSP so

let's say that and a lot of people ask things related to this let's say that you have this program now which is a bootstrapped fuse shot react agent built in DSP as a tool that you can use in your life chain you know conversational pipeline but it's kind of expensive because it's going to talk to GPT 3.5 or gpt4 or whatever it is you know several times in each query how can you make this efficient and how can you make sure this

adapts to the actual queries people ask so our solution for this is you deploy your agent and in particular you deploy this tool in front of users you gather a bunch of questions from them these questions don't have labels they don't have answers but you know you could still you could still gather them and and save them and then he could just say hey ESP can you please compile my program that we built here and use these

user questions and I'd like to give get to get that you know I'd like you to use in this pipeline maybe the default language model that you have which might be configured to be T5 large you know in on on on GitHub the default is Ada open AI Ada so in you know we've been playing for with this for a while and you know if you compile the you know a DSP program that does multi-hop search into T5 large you match

the quality of retrivo that you're getting from you know the DaVinci 3.5 and you only lose a very small kind of margin in terms of The Final Answer accuracy but you don't need to go to open a ISO anymore and you know this whole thing it requires like say hand labeled examples to build the initial sort of bootstrapping of this and so lots of opportunities here in terms of being able to build your own self bootstraps

retrieval Tools in something like line chain so you know I'll I'll stop here certainly excited about Integrations with what people are doing with in terms of Colbert just as an initial you know Standalone general purpose thing and also with DSP programs that individually Implement you know you know pipelines for retrieval in a more task specific or task aware manner so Omar can I ask a quick question about the the DSP stuff so the so the demonstrate step is that kind of

equivalent to coming coming up with like few shot examples dynamically basically like you give it yep okay and so and and so is that is it like like could you could you do the demonstrate step get those few shot examples save those and like have those as as like or is it dynamically generated every time as you go through so this is entirely program dependent I encourage people to read the paper for you know different we described like design patterns for

most tasks if you're just getting started yeah this could be a frozen set you just generated three examples and they're fixed but you could also Imagine fancier things where you haven't you got a new question you have a bunch of questions in your old database and on the Fly you retrieve similar examples you have a k n thing in in Lang chain you're going to use it to achieve some other examples and then annotate those on the Fly so

that the model the language model can see similar examples being answered and leverage those there yeah I guess I should keep my answers short so that we have time for oh yeah so I wanted to grab some questions from the audience and make sure to get a chance to get you know Joe and Omar in in conversation with each other so I'm gonna grab this one from Zaire so we talked a lot about like q a

that's been the task that people have used done retrieval for but the tasks the thing that people are really excited about right now especially in The Lion King Community is around agents and one use of retrieval for agents is creating this sort of like hacky long-term memory and allowing for self-improvement so what of the like approaches that we've seen so far in this talk or that maybe are you're aware of Joe and Omar do you think our maybe most useful for helping

with this with this problem of improving the memory of like an agent maybe we'll start with with Joe yeah I I I'm not that familiar with the terminology used around long-term memory short-term memory so it's it's it's my understanding was that long-term memory was for about the knowledge database and not the intermediate steps but I might be wrong so I'll pass that

question maybe Omar is more into the age yeah I actually also don't know the specific Lang chain terminology here but but in terms of self-improvement there are two paths in general there are models and I think this is the one that's a lot more common in the agent world is kind of per example or post-hoc self-improvement so you

you're trying to do stuff you make a mistake you reflect and then you do a better job next time

that's great that's very versatile and again you know like all these zero shop agents they allow you to do things that like very recent not not you know just a week before length chain people wouldn't have thought were possible you know in terms of the abstractions that generalized react and other things but being reactive to the extent of reflecting per example might mean silly things like you make the same mistake every time and then you reflect on it

and then you realize you made a mistake and then you keep fixing it so that's a little bit expensive and maybe kind of ill-advised and this is this is where the abstraction that we have around demonstrations in DSP among other sort of things that fit up fit together say well you know if you can collect some of these a couple of examples in advance then you can sort of essentially bootstrap your pipeline on them in advance and it could avoid these sorts

of Errors with some you could you could sprinkle in a little bit of rules so like the the the intuition and DSP is that you're you're building up your understanding of the task as you go and if you could just say hey I want to handle this exception where you know I'm trying to search and I keep searching like too many times in a row or like I try to answer before searching or whatever it is that happens in your

agent because as an exception for in the code or add like an if statement for it in the code or whatever it is Rerun recompile you get new demonstrations you build up a new model underneath if you want to compile the way and you know it gives you gives you that path for self-improvement that is maybe a little bit less reactive but you could certainly use it with agents or you know it it it fits in the bigger

pipeline because this is something I don't anticipate I don't see you seeing doing this at every little edge of your pipeline this is probably going to be what you want to reserve like the research part of your brain and research part of your team when you're focusing at the key kind of core differentiator of your application maybe that search maybe that is some generative component whatever it is at least that's that's my you know my the way I think of this

yeah I think maybe yeah Harrison can maybe comment on this a little bit but I think with with yeah the self-improvement I think you're right that like finding ways to make that self-improvement persist from one conversation to another is huge and DSP seems like a promising way to do that I think in terms of the retrieval and long-term memory side for agents the way it normally looks is that like the amount of information that you're searching over is going to generally be

smaller right it's like the memory of a single agent that's more like the scale of the amount of information a person consumes on on the internet so we're talking hundreds of documents thousands of documents rather than Millions and then you also are maybe doing some annotation of that at the same time with the language model so it's not just a document it's a document with additional metadata which is a pattern that foolish sailor brought up in one of

their questions about adding llm like adding Rich metadata to documents that's also in natural language I think those are kind of the the questions that are on people's mind for agents what if I have like a relatively small amount of documents where I want to achieve like really really high Precision and like maybe reasonable recall I don't care as much about like turbo scale and yeah and I'm doing that in a context where there's loms in play so

there's lots of opportunity for like metadata enrichment some of this might be previous outputs of the llm not necessarily internet data there's a little bit more control there so what I would say here is you know I I think there are very nice abstractions and length training that I'm familiar with here like you know the like you guys have the the essentially mapreduce thing if you have data that's that small I imagine that's just a very natural thing

you can do if the data gets to a level where that's too expensive you could essentially use the compile function in DSP to replicate that behavior but with the language model that got fine-tuned to be a tiny thing that you know understands the specifics of what you wanted to do so you don't need you know you don't need the full power of gpt4 to you know do that specific map reduce process but but if you just if you try

to prompt flan T5 it it can't really do it reliably so the compiler will do is hey let's simulate it a little bit what gpt4 would do on a few unlabeled examples and you know teach it to the to a small model that updates adaptively as we change our programming and so that's one thing you could do there alternatively I think Colbert V2 indexing of small data sets is a you know it's an exciting Direction it's certainly makes it

even less of a concern the fact that you have a lot of vectors to store I don't think it's a concern to begin with unless you have like many billions of documents at which point it becomes a bit of a challenge but if you have hundreds or thousands definitely try this out like index it you know you might not even you know need a very powerful GPU for indexing you do need a GPU for indexing now just because the

code assumes that you have one but fundamentally if people pick this up you know you could run it on on CPU with fast birth implementations search is CPU friendly though like you don't need gpus for search just for indexing right right yeah and Joe any thoughts about this specific context of long-term memory where you've got like maybe smaller amounts of documents richer richer metadata more control and I think this the mechanisms are the same right you

want to retrieve over this and as I said the I think the hybrid combinations and and combining and also taking whatever text is there and the metadata and the generations and that you actually can store this data in any kind of store that will allow you to store both the text data and the vector representation if it's tensor data like in case of Colbert or is the single Vector model and then it's just that you don't have to introduce any

kind of approximate Vector search you can do exact search on those smaller data you don't you know you can run the full callback model and maybe even you know you can run a full cross encoder model if you use like a mini language model you know because they are you know they also I think they're overlooked in in the current landscape but they do have some advantages as well right okay if you have gpus you don't need to store

anything you can

retrieve simply and then re-rank using a GPU and that's why the investment team has been focusing on adding GPU capacity and investment clusters like for doing that because I think cross encoders it's really something a very easy way to lift accuracy even if you're running on a simple bm25 retriever yeah so there are a lot of options and and informational achievable and working in search because there's so many fascinating problems like sub problems of search query understanding document

understanding ranking you know diversity someone mentioned here in the chat you know what do I do for diversifying the resource that I put into the to the language model so there's so many aspects going on right and and now we just touched on you know how to how to evaluate that ranked list in the context of Agents but there's so many things going around and yeah and I'll I'll share my my presentation on Twitter later on and I also share some research

this is a free online courses on information material if you want to learn from experts so so people can get up to date on the general stuff like this Joe Joe it sounds like there's so much going on that we might need to do a rerun of this basically that's what you're saying right yeah that's what I'm saying yeah and so we didn't get a chance to answer everybody's questions but I know Harrison and I and Joe are all on

Twitter and I've seen both of them frequently having interesting conversations with people who have questions Omar is as well Omar's as well yeah Omar as well great interaction Perfect all right oh that's your Twitter handle late interaction yeah yeah it's actually a it's actually a fun on the on The Late Show with Colbert oh got him incredible well thanks for your Colbert Report on on the late interaction and multi-factor retrieval and thanks to Joe for the overview and showing us some of

the stuff that vespa's building working on and as always thanks to Harrison for organizing the Lang chain Community putting Lang chain together and thanks to Charles for hosting this and and keeping everything running smoothly and I'll just say we I run a little like online a little online education stuff the full stack llm boot camp full stack deep learning course Joe gave our information retrieval lecture the the thumbs up the coveted

Joe bergum thumbs up so yeah definitely check it out fantastic resource I can't believe you gave away that for free I mean Jesus Christ I was like it was fantastic I mean go watch it you know go to that link watch all that material you know very fantastic content yeah really good stuff thanks Joe all right well thanks everybody for coming and we'll see you at the next one of these yes