Beyond the Basics of Retrieval for Augmenting Generation (w/ Ben Clavié)

48 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=0NA5QG3087G
https://www.youtube.com/watch?v=0nA5QG3087g

SUMMARY

Ben Clavier discusses the basics of retrieval-augmented generation (RAG) systems, emphasizing the importance of understanding the components involved in building an effective RAG pipeline. He introduces concepts such as B-encoders, cross-encoders, and the significance of metadata filtering, while also addressing common misconceptions about RAG.

- *RAG is not a standalone system but a combination of retrieval and generation components that work together.*
- *The compact MVP for RAG involves using vector search for document retrieval and can be implemented with minimal code.*
- *B-encoders create embeddings for documents, while cross-encoders evaluate the relevance of document-query pairs.*
- *Incorporating traditional keyword search methods like BM25 alongside vector search can enhance retrieval accuracy.*
- *Metadata filtering is crucial for refining search results based on specific attributes, improving the relevance of retrieved documents.*
- *The presentation highlights the importance of continuous monitoring and improvement of RAG systems.*
- *Clavier encourages the use of open-source tools like the Ra Rouille library for implementing RAG effectively.*

Ben clavier is you know one of the cracked researchers who work at answer AI you've heard from several researches from answer already in this conference Ben has a background in information retrieval amongst other things and has he has an open source package called ra rouille which you should check out it also comes from a deep background in information retrieval and brings that to Rag and he also one of the clearest thinkers on the topic but yeah I'll hand

it over to you Ben to kind of give more color to your background anything that I missed and yeah we can just jump into it okay let's go so I think that's pretty much the key aspect of my background you pretty much read the slide out so I do yeah I do andd at with like JY you've seen jono in this course and there's like a lot of other awesome people we a distributed R&D lab so we do

AIR research and we try to be as open source as possible because we want people to use what we build prior to joining answer I did a lot of NLP and kind of stumbled upon information retrieval because it's very very useful and everybody wants information retrieval it's more of like clarifying what information retrieval is which I hope today's talk will help and yeah my SOI claim to fame or claim to moderate Fame at least is the ratu

library which makes it like makes it much easier to use a family of models called C Bear which we will very briefly mention today but w't have time to go into detail but hopefully like if you want to know more about that like do feel free to ping me on Discord I'm generally either very responsive or you need to ping me again pretty much how work and I also maintain the reun C library which we will discuss in one of

the later slides and yeah if you know me or want to follow me I want to hear more about why I do it's pretty much all on Twitter I'm not on LinkedIn at all I'm just everything go through Twitter a lot of memes and CH po but some very informative stuff once in a while so so yeah yeah and so let's get started with what we're going to talk about today and so it's only half an hour so we're not going to

talk about a lot I'm going to talk about why I think are the like cor retrieval Basics as they should exist in your pipelines because rag is a very nebulous term and that will be the first slide and ham will be very happy about that slide I think but rug's not a silver bullet rug's not a new thing from December 2022 rug's not even an system will cover that but I think it's very important to like ground it a bit when

we talk about drug because it means a lot of different things to different people then we will cover B what would call the compact MVP which is what most people do when they are starting out with rag It's actually an example from Jeremy and it's like the simplest possible implementation of rag as in just using Vector search and then the other topics are basically thing that I think you should have in your rack pipeline as part of

your MVP and I'll show that like there's a lot of scary Concepts because they're all big walls like bu encoder cross encoder tfidf bm25 filtering that sounds like a lot but then I'm going to try and show it but they're very simple Concepts and you can have pretty much the same MVP by adding just 10 lines of code but using like Bally state-ofthe-art retrieval components in every bit and the bonus which I don't think we'll have time to cover when I try this

again was talking about C Bear because I like talking about colar so I might do it at the end if we have some time but I might not and yeah that's it for the agenda and then I also think it's important to have the cter agenda which is what we won't be talking about today because those are just as important for rag but they're not what I would put in the very Basics and here very much about the

basics so one of them is how to monitor and improve rag systems because Rags are systems and they're living systems and they're very much things you should monitor and continuously improve on I think Jon covered that quite while in the talk yesterday or last week yeah last week so I would invite you to watch that and watch Jon and Dan's upcoming course if it does materialize evaluations they're also extremely important but we won't talk about them

at all today but I know that Joe we talk about them at length in this talk benchmarks and paper references so I'll make a lot of claims that you will just have to trust me on because I don't want to like have too many references or too many like academic looking tables and they trying to keep it quite Lively and a I won't give you a rundown of all the best performing models and why you should use them I won't talk about

training data augmentation Etc and I won't talk about all the other cool approaches like play call Bear in details because they go beyond the basics but those are all very important topics so if you're interested do look up there's a lot of good resources out there do feel free to ask me and with that let's get started with the rant which is my favorite part so this is a thing that H been doing on Twitter recently as part of his flame

posting campaign I say which is basically there's so much in AI so much especially the LM world that uses wordss that are like a lot scarier than they need to be and rug's probably that because to me when I hear retrial generation or rag it sounds like that's an end to end system that's a very definite set of component that's a thing that works on its own and it's not it's literally just doing retrieval to put

stuff into your prompt context like before your prompt or after your prompt you want to get some context you're doing retrieval but that mean that's not an n2n system despite what Jon will have you believe on his Twitter is not created it but it does make a lot of money from it and it's basically just the act of stitching together R Revol so the are part of rag and generation so the G part of like to ground the lat so

like you want to your generation to be grounded to use some context so you're doing retrieval on whatever documents you have and pass it to your LM but there's no magic going on it's very much like a pipeline that take the output of model A and gives it to model B the generation part is what's handled by the large language models and good rugs actually three different components it's a good retrieval pipeline it's a good generative model and it's a good

way of linking them up so it can be formatting your prompt or whatever and it's very important to think about it when you're saying my rag doesn't work you need to be more specific like my rag doesn't work is the same I say like car doesn't work it's like yeah but something specific is broken you need to figure out what is it the retrieval part is the LM struggling to make use of the context Etc there's a lot of fat cases

there and with that being said let's look at what like the compact MVP is so that is basically what you will see I think if you've read any medium block post about the Advent of flag in ear 2023 that's the pipeline that everyone used and that's also because the easiest pipeline to bring to production it's very simple you have a query you have an embedding model you have documents the documents get embedded and pulled into a

single Vector then you do cosine similarity search between the vectors for your query and for the documents and that gets you your result that gets you your score and this is a bit of a teaser For an upcoming slide when I say this is called the B and cod approach but just so you get the term in mind and I'll Define it because that's one of those things that is like a scary term that's actually very very simple when you break

it down but first let's look at what this actually means in code this whole pipeline so the first thing you want to do is load your model then you get your data you encode it you store your vectors then you get your query you encode it and then here we use n you do a cosign similarity search it a DOT product between normalized vectors to get the most similar documents and the document that are similar to like

the documents who edding are similar to your qu edding is what you'll consider as your relevant documents here and that's pretty much it that's modified from something that Jeremy did to Showcase how simple rag actually is in his hackers guide to LMS but that's what you want to do to retrieve context in the simplest possible way and you will have noticed that there's no Vector DB in this this is all n arrays and this is all n p because

when you use Vector DBS the huge point of using a vector DB is to allow you to efficiently search through a lot of documents because what a vector DB does generally not all of them but most of them wrap stuff like hnsw IVF PQ which are indexing types and what that allows you to do is to find and retrieve relevant documents without having to compute cosine similarity against every single document it like tries to do an

approximate search or an exact search this is not something that you need if you're embedding like 500 documents like your CPU can do that in milliseconds you don't actually need a vector DB if you're trying to go to the simplest possible stage but if you wanted one it would go right here on the graph like right after you Ed your documents you would put them in the vector DB and the second thing I think to discuss about this like this tiny graph

is why am I calling embeddings B encoders because that step that I call B encoder you will have seen a lot of times but you will already see it generally called embeddings or model and B coder is the term that the IR literature uses to refer to that and it's simply because you encode things separately like you do two encoding stages so it's a bu encoding and that's used to create single Vector representations where you precompute all your documentation so

when you're using buy encoders you encode your documents whenever you want like when you're creating your database when you're adding documents those get uncoded at a time that's completely separate from inance and then only at inance you like in the second aspect of this column will you embed your query to compare to your precomputed documentary presentations so that's really really computationally efficient because at in France you're only ever encoding one thing which is the query and everything

else has been done before and so that is part of why it's done that quickly and I did want to take a slight break because I can see there are questions but they're not showing up on my screen so there I need on like this quick MVP done yeah let me look some of the questions I'm going to give you a few of them you can decide whether you want to take them now or later so we got one

it's a 7,000 query a 7,000 question and answer data set can I optimize rag to accurately retrieve and quote exact answers will also effectively handing queries it's slightly different from the original data and there's actually two parts of that so one is to quote the exact answer to something about the is not the information retrieval part but it's rather just like what do you tell the LM to do but the information retrieval part is probably well you see

so go I will actually cover how to better deal with out of context things in like upcoming slide it's not yeah why do you keep going none of these questions now yeah perfect yeah okay so the next one is if that's very computationally efficient there is an obvious trade-off here and that is your documents are entirely unaware of

your query and your queries are entirely unaware of your documents which means that you're very very like subject to how it was trained is

basically if your queries look a bit different from your training data or if like if there's very very specific information that will be in certain documents and not other sometime you want to know what how the query is f you want to know what the quer is looking for when you're encoding your document so that it can like kind of paint that representation represent it more toward information that you're interested in and that's done with what we call

reranking so reranking is one of those scary stages that we'll see in your pipeline and the most common way to do reranking is using something we call cross encoder and cross encoder is another one of those scary wordss like B encoder that you feel should be like a very Advanced concept but it's actually very simple this graph here represents the whole difference between them the Bon coder is basally this two column system that we describe where documents

get uncoded in their Corner queries get uncoded in their own corner and they only meet very very late like you only do coine similarity between vectors but the documents never see in the query and VI the cross encoder is different the cross encoder is a model that will take your document and your query together so you're going to give it both your document or like a series of documents depending on the type of model but to

keep it simple we do it one by one so you always give it a qu document pair and you put it through this cross encoder model which is effectively a classifier with a single label and the probability of the label being positive is what your model considers as how similar the documents are or how relevant it is this is extremely powerful because it means that the model knows everything about what you're looking for when it's uncoding the document and can give you a

very accurate score or at least a more accurate score the problem is that you can see how that wouldn't scale because it's not very computationally realistic to compute this like query document score for every single query document pair every time you want to retrieve a document say you've got like Wikipedia embedded you've got I don't know like 10 million paragraphs you're not going to compute 10 million scores through a model so like using 300 million parameters for every single document or

you you would eventually return something and it would be a very very relevant document but it will also take 15 minutes which is probably not what you want in production so you probably also have heard or you might also have heard if you're really into retrieval or not hell at all if you're not into retrieval of other R ranking approaches like rank GPT or rank LM using LM to rank documents has been a big thing lately for people

really into retrieval you will know of mon 5 Etc so those are not cross encoders but that's not really relevant to us because the core idea is the same and that's basically what we always do with reer ranking in a pipeline you use a powerful model that is computationally expensive to score only a subset of your documents and that's why it's reranking and not ranking because this can only work if you give it like I don't know 10

50 not more than that documents so you always have a first sted retrieval which here is our Vector search and then the ranker does the ranking for you so it creates another list there's a lot of ways to try those models out some of them have an API base so it's just an API call to coh or Jr some of them you run your machine if you want to try them out and this is basically the self promotion moment I do

maintain as Library just called rankers with a QR code here where it's basally unified API so you can test any R ranking method in your pipeline and swap them out freely and that's what your pipeline looks like now it's the same with just that one extra step at the end where you rerank things before getting your results so we've added reranking but there's something else that's missing here and that's something actually addresses the first question at least

partially is that the semantic SE V eddings is powerful and I'm not saying don't choose vectors vectors are cool like models are cool deep learning is cool but it's very very hard if you think about it because you're asking your model to take I don't know 512 tokens even more if you're doing long context and you're like okay put all of this into this one vector we're just using a single Vector you've got like I don't

know 384 1024 at most flots and that must represent all the information in this document that's naturally lossy there's no way you're going to keep all of the information here and what you do when you're training on embedding is that you're teaching the embedding to represent information that is useful in their training so the model doesn't learn to represent all of the document information because that's pretty much impossible since eddings are essentially a form of compression what the model actually

learned is to replant the information that is useful to the training queries so your training data is very very important here it's like representing the documents in a way that will help you use the queries in the weather phrase in your training data to retrieve a given document so when you use that on your own data it's likely that you're going to be missing some information or when you go slightly out of distribution there's another thing which is humans

love to use keywords especially if you're like going into the legal domain the biomedical domain anything specific we have a lot of acronyms that might not even be in the training data but we use a lot of acronyms we use a lot of like very Advanced Medical worlds like people love dragon people love to use technical WS because those are very very useful and that's why you should and I know it sounds like I'm talking from the

70s because that's actually a method from 70s but you should always have keyword searched in your pipeline you should always also have full text search on top of like anything that you do with vectors and keyword search which you can call full text or like tfidf bm25 it's powered by what we call tfidf which is a very basic NLP concept that essentially stands for term frequency inverse document frequency and it assigns every single word in a

document or a group of words because sometimes we do them 2 by two or 3x3 even it gives them a weight based on how rare they are so like a that appears everywhere like V or a as a very very small weight and a world

that's like highly specific to certain documents at a very high weight and the main method to use tfdf for retrieval is called bm25 which stands for best matching 25 it was invented in the 70s it's been

updated since then but it's basically just been iterations of it and you'll often hear IR researchers say that the reason that the field is not taken off like NLP has or computer vision has is because the Baseline is just too good the still competing with bm25 even though it's been 50 years now my God it's been 50 years yeah so bm25 existed for like basically my anti lifetime before my birth and it's still used in production PIP plan today that's how

good it is and the good thing is it's just work counting with a M with like a waiting formula so the compute time is virtually unnoticeable like you can add that to your pipeline you will absolutely never feelit it and I know I said I wouldn't add anything from papers but I feel like like because I'm making a very strong claim that this method from 70 is strong I should add a table and that the table

from the be paper which is the retrieval part of MTB which is basically the main embeddings Benchmark and they compared it to a lot of models that were very popular for retriever like DPR and very strong Vector retrievers and basically you can see that unless you go into very overtrain eddings like E5 BGE the M25 is competitive with virtually all deep learning based approaches at least at the time of the paper which was only just three years

ago we now have embeddings that are better but we don't have any embeddings that are better to the point where they're not made better by being used in conjunction with bm25 so knowing that this is how you want your pipeline to look you'll notice that there's now a whole new pathway for both the query and the documents were on top of being uncoded by the embedder also encoded by tfidf to get full text search and that will help you like

retrieve keyword ET humans use keywords in qu all the time it's something you should do at the end you will combine the scores it's you can do that in a lot of way I won't go into too much details but what a lot of people do is give a weight of 0.7 to the cosine similarity score and 0.3 to the full Tex s but I'm pretty sure we could do a whole talk for now on different methods

of comparing that okay I do have five more minutes so the last one that you want to add to a simple pipeline the thing that I think really completes your MVP Plus+ is using metadata and using metadata filtering because academic benchmarks don't because in academic benchmarks documents exist mostly in a vacuum like they don't exist in the real world they're not tied to a specific company Etc when you using rag in production it's very very rare that

someone comes to you and says these documents came to me in a dream and cut them like they came from somewhere they they've been generated by a department they've been generated for a reason they might be old Excel sheets or what but they like they have business sense or they have like in context sense and the metadata is actually sometimes a lot more informative than the document content especially in rag contexts so if you take

the query here

which is can you get me the cruise division financial report for Q4 22 there's a lot of ways in which this can go wrong if you if you're just looking at it from like the semantic or even using keywords aspect So when you say when you see like this the model must capture the financial report so you the model must figure out you want a financial report but also Cruise division Q4 and 2022 and embedding models are bad at numbers so you might

get a financial report but maybe for another division or maybe for the cruise division of 1998 it's very hard to just hope that your vector will capture all of this but there's another fail case which will happen especially with weaker LMS in that if you just have like top top Cas in top five documents and you retrieve the top five documents for your query even if your model is very good if you just let it retrieve the top five

documents no matter what you will end up with financial reports at least five of them and there's most likely only one for Q4 22 but at that point you're just passing all five to the model and being like good luck use the right one which might confuse it especially because tables can be hard Etc and so I'm not saying that your vector search will fail but statistically it will like in most cases it will fail and

if it don't fail for the square it will fail for a similar one but that's actually very very easy to mitigate you just have to like think outside of the vector and just use more traditional methods and you can use entity detection models and one that's very good for this is gler which is a very recent model that does basically zero shot entity detection so you give it arbitrary entity types so like document type time period and the

department and this is like a live thing of gler you can run the demo on the bottom but here we just extract financial report time period and department and when you generate your database for rag all you need to do is basically specify the time period so when you get an Excel sheet you'll just pass the name for it or pass the date in it and give metadata 2024 Q2 Q4 sorry 2022 Q Q4 okay mix up there

and then you just need to ensure that this is stored alongside your document and at query time you can always pre-filter your document set to only query things that make sense so you will only query documents for for this relevant time period so you ensure that even if you give your model the wrong thing it will at least be the right time frame so it can maybe try and make sense of it and with this final component this is

what your pipeline looks like you can see the new component here which is metadata filtering which doesn't apply to queries s is go right through it the documents get filtered by that and we won't perform search on documents that will not meet the metadata that we want and okay I do agree that this looks a lot scarer than the friendly one at the start which just had you better and then coign similarity search and the results but it is actually not very

scary this is your full pipeline this implements everything we've just talked about it's about 25 lines of code if you remove the commands it does look a bit more un friendly because there's a lot more like moving Parts I

think there's a lot more steps but if you want we can just break it down a bit further and so we use lens DB for this and this is not necessarily an endorsement of L DB as a vector DB

although I do like L DB because it makes all of these components which are very important very very easy to use but I don't I try not to take side in like the vector DBS because I've used weav I've used chroma I've used L DB I've used pen con they all have their place but I think L DB if you're trying to build an MVP is the one you should always use for MVPs right now because it had those

components built in and here you can see just how easy it actually is so we still load the B en coder just in a slightly different way same as aler we Define our document metadata here is just a string category it could be a time stamp it could be just about anything then we uncode all of our documents just like we did previously and that's here we've created so it's not an index this is still a hard search this is not an

approximate search then we create a full text search index which is generating those tfidf so why I mentioned before we give away to every single term in the documents then we load the ranker here we're using the coh ranker because it's simple to use an API and at the very end you've just got your query and your search where we restricted to the category F films so we will only ever search into the document that's about a film not about an author

not about a director we get the top 10 results and we just have a quick ranking step and that's pretty much it we've taken the pipeline at the start which only had the B and coder component to a pipeline that has the Bon color component metadata filing full Tech search and a reor at the end so we've added like basically the four most important components of retriev into a single Pipeline and it really don't take much more space in your code

and yeah that is pretty much the end of this talk so there's a lot more to cover in rag this is definitely not the full cor flag but this is the most important thing like this is what you need to know about how to make a good pipeline very quickly all the other improvements are very very valuable but they have a decreasing cost effort ratio this takes virtually no effort to put in place definitely worth learning about sparse

methods multi Vector methods because they're very adapted to a lot of situations C Bear for instance is very strong out out of domain SP is very strong in domain you should watch Jon's talk about rag systems and Jo's upcoming talk about retrial evaluations because those are by CLE trifect the most important things and yeah any questions now haml and I were just messaging saying we love this talk and like it's just everything is presented so clearly

so we've also got quite a few questions I happy my favorite talks of for so you not big favorites but yeah thank you go ahead okay questions you were gonna pick did you have one that you were looking at already D I can try yeah we've got one that I quite like in the way that you fine-tune your buy encoder model affects how you should approach fine-tuning for your cross encoder and vice versa yes I don't think I can give like

a really comprehensive answer because it will really depend on your domain but you generally want them to be complementary so if you've got if you're in a situation where you've got like the computer and the data to F both you always want your buy encoder to be a bit more loose like you want it to retrieve potential candidates and then you want to trust your ranker like your cross encoder to actually do the filtering so if you're going to use both and have

full control over both you might want to P tunate in a way that will Bally make sure that your topk candidates can be a bit more representative and Trust the ranker let me ask this wasn't a an audience question but related question you showed us where the when you choose questions to feed into the ranker that's sort of a weighted average of what you get from the tfidf or

bm25 with what you get from the just simple Vector search what do you think of as the advantage or disadvantage of that over saying we're going to take the top X from one cat from one of the rankers and the top X from the others and that way if you think one of these is is for some questions especially bad you have a way of short short circuiting its influence on what could sent to the the

rer yeah I think that also makes complete sense and that's another that's a cop out answer I'll use a lot but that also depends a lot on your data like a lot of the time you want to look at what's your actual context and how it's actually being used because in some situations that actually works better like especially if you work with biomedical data because there's so much like specific documents it's quite often the embedding won't be that amazing on

some questions so you just want to take the top five from both and get the rear car to do it the rear car is qu aware so it's a perfectly valid approach to combine them that way you want to pick a question HL yeah I've been looking through them you guys been okay Jeremy's asking can we link get a link to the code example your slides in Maven we can also can I share your slides in Discord as well Ben yes just

yeah I'll go ahead and share the slides on Discord yeah and I'll share like the GitHub just for the code examples of phot and I'll embed the link to the slides in Maven for people who watch this talk some point deep into the future and who might lose track of it in Discord there's a question somewhere in here I'll find in a moment but we got this question for Jason the speak and then the speaker just before you Paige

Bailey said rag you know in the world of million token context lengths is not going to be as important what's your take on the relative importance of rag in the future so I'm still very hopeful about rag in the future and I think I see it as some some sort of like so yourm to me is like your CPU and your context window will be your RAM and so like even if you've got 32 gigs of from nobody's ever

said yeah throw away your hard drive you don't need that like in a lot of context you'll still want to have like some sort of storage where you can retrieve the relevant documents having to use a long context window is never going to be a silver bullet just like rag is never a silver bullet but I'm actually really happy because it just means I can retrieve much longer documents and get like more efficient rack systems because

to me it's a bit of a trade-off where if you've got a longer context it just means you've got a lot more freedom with how quick your retrieval system can be because if it if you need to use top 10 not at top 15 that's fine you can fit them in whereas when you can only fit the top three documents you need your ritual system to be really good which might mean really slow we had a question

from Wade Gilliam what what are your thoughts on different chunking strategies I probably don't think about chunking as much as I should I am very hopeful for future Avenues using llms to pre chunk I don't think those work very well right now just in my test I've never been impressed but also I do tend to use C Bear more often than b coders and C Bear is a lot more resistant to chunking so it's something that I don't care about as much but

generally I would try to so my go-to is always to chunk based on like around 300 tokens per chunk and try to do it in a way where you never cut off a sentence in the middle and always keep like the last 50 tokens and the next 50 tokens of the previous and next chunk because information overlap is very useful to give content like please don't be afraid to duplicate information in your chunks I have a question about

the buy encoder do you ever try to fine-tune that using some kind of like label data to get that to be really good or do you usually kind of use that off the shelf and then use a ranker and you know how do you usually go about it or how do you make the tradeoff so again context dependent but if you have data you should always find all your unod be the buy anod the cross encoda I think call be because single

Vector you can get away with not fing for a bit longer because multi Vector sorry you can get away with not fing for a bit longer but if you have data it's all about like Bally the resources you have so in this talk we're doing an MVP is something you can put together in an afternoon if your company says you have $500 spend 480 of that on open a to generate synthetic question and F your your encoders that will always get you

better results like always F turn if you can and so I saw a couple questions about fitting C Bear in and I'm using present executive decision to answer those so call Bear in this pipeline some people use it as a ranker but then that's not

optimal that's very much when you don't want to have to change your existing pipeline if you were to design the pipeline from scratch and wanted to use colar you would have it instead of the buy and coder and it will perform basically the same role as the buy and coder which is first stage retrieval and if you wanted to use C Bear and especially if you don't have the budget to find tune and need a reranking step sometimes it can actually

be better to use SC bear to re ranker still because the multi Vector approach can be better at like capturing keywords Etc but that's very context dependent so ideally you would have it as short by encoder for a lot of people here probably aren't particularly familiar with coar well bear can you give The quick summary B yeah sorry I got carried away because the question so colar is an approach which is effectively a buy and coder but instead

of CR cramming everything into a single document into a single Vector you represent each document as a bag of embeddings so like if you've got 100 tokens instead of having one big 124 Vector you will have a lot of small 128 vector one for each token and then you will score that at the end you will do the same for the query so if your query 32 tokens you will have 32 query token and for each query token you will compare it

to every token in the document and keep the highest score and then you will sum up those highest scores and that will be the score for that given document that's called Max similarity and the reason that's so powerful is not because it does very well on data it's been trained on actually like you can bat it with a normal B coder but it does very well extrapolating to out of domain because you just give the model so much more

room to replant each token in its context so it's much easier if like you're in a non-familiar setting you've not compressed as much information and I do have self-promotion I do have a pretty cool call Bear thing coming out later this week to compress the C Bear Space by reducing the tokens that actually needs to save by about 50 to 60% without losing any performance so that's a bit of a is well but look forward to the blog post if you're

interested and to find the blog post you suggest people follow you on Twitter or on yeah definitely follow me on Twitter because Twitter is pretty much the only place where you can reliably reach me someone's asking what are some good tools to fine-tune embeddings for retrieval would you recommend ra roui or anything else like what's your i' would recommend sentence Transformers especially with the like 3.0 relas recently it's now much much friendlier to use and it's basically there's no

need to reinvent the wheel they've got all the basics implemented very well there so sentence Transformers question from Divia can you give any pointers on how one fine tunes their embedding model sorry can you repeat that I could yeah the question is can you give any pointers or describe the flow for when you fine-tune your embedding model okay so that's probably a bit more involved than this talk but essentially when you find in your embedding model

what you'll want is queries you need to have queries and you need your documents and you're going to tell the model for this given query this documents relevant and for this given query this documents not relevant because sometimes you use a triplate loss and a triplate loss is what you will do when you have one positive document and one negative document and you'll kind of be teaching the model this is useful this is not useful and I'm not going to go down too

much because this Rabbit Hole can take you quite far but sometimes when you have Tri plates you also want to use what we call hard negatives which is you want to actually use retrial to generate your negative examples because you want them to be quite close to what the positive example is but not quite the right thing because that way you teach the model more but was actually useful to answer your query so the workflow is

probably as always look at your data figure out what kind of queries your user will actually be doing if you don't have user queries go into production write some write some queries yourself and give that to an LM generate more queries and you can have a pretty solid ritual by plan like that someone's asking in the Discord and I get

this question all the time is please share your thoughts on graph rag I have never actually done graph rag

I see this mentioned all the time but it's not something that has come up for me at all so I don't have strong thoughts there I think it's cool but that's pretty much the full extent of my knowled someone's asking okay when when you have long context Windows does that allow you to do something different with rag like retrieve longer documents or do any other like different kinds of strategies than you were able to before or does it

change anything how you go about this yeah yeah I think it's a bit why I mentioned before to me it changes two main things one is I can use longer documents which means I can use longer models or I can like Stitch chunks together cuz sometimes sometimes if your retrieval model isn't very good at retrieving long documents which is often the case you might just want if I get a chunk from this document give the model

the full document like if I just get a chunk from it pass the full context and you just hope the model is able to read it and if you've got a good long context model it can so it changes how you decide to fit the information into the model and then the other aspect is like I said it changes the retrieval overhead because if you need to be very good like I was saying if you need only the top

three documents to be relevant you're going to spend a lot of time and money onal pipeline if you're like oh as long as my recall at 10 or my recall at 15 is good that's fine you're you can afford to have like much lighter models and spend a lot less time in resources on retrieval like there's a lot of diminishing returns in retrieval when getting a good recall at 10 so like recall at 10 is How likely you are to

retrieve the relevant document in the first 10 results is generally very easy recorder to 100 is very very easy and then recall at five is getting harder and recall at three and recall at on are like the really tough ones because a lot of the training data is noisy so it can even be hard to know what a good record one is the longer context makes that irrelevant and that's why it's great for someone's asking I don't even know what

this means what's your view on hide versus react versus step back I've only used react out of those and so those are like agentic systems of function coding so it's like to give yourm the ability to call tools at this react is I don't have strong thoughts on those in the context of R reval so I can't really answer the question yeah I think I would occasionally use react from the model to be able to trigger search itself but I

think that's still an open area of research and I think Griffin from an is also in the in the chat and is very interested in that is basically how do you get a model to tell you that it doesn't know because sometimes you don't need retrieval the model already knows sometimes you do need retrieval but that's still a very open question like how do you decide when to search so no strong thoughts there yet there there may may you may or may

not have a good answer for this one is there an endtoend project open source project that someone could look at as a way to see or evaluate the difference in result quality when they do result from just buy encod or MVP and

compare that to the final compact MVP Plus+ that you showed no actually that's a very good point I don't think there's one that systematically go through every step and that's probably something that I would

like to build at some point of f in one because just like most things in retrieval everything is kind of conventional wisdom like you've seen it bits and pieces in a lot of projects and you just know that that's how it is but unless you dig deep into the papers or like do it yourself it's quite rare to find very good resources showing that related question do you have a tutorial that you typically Point people to on

fine tuning fine tuning their encoder that would be the sentence Transformers documentation but it's not the friendliest tutorial so that's a half answer that's why I would punch you to but it's still a bit like hard to gain to sadly

wait is asking if you have goto embedding models I think my goto these days when I'm demoing something in the Korea one because it's nice to be able to work with an API it works really well it's cheap but other than that no I would just call Bear if I'm using something in my own pipeline I would just like multi vectors but it really depends on the use case because you will often find that like some things work well for you and some

things don't I do have strong opinions on not using so if you go to the MTB leaderboard which is the embedding leaderboard right now you'll see a lot of LMS as encoders and I would had advise again that because the latency is on four it don't need 7even billion parameters to uncut stuff and at least some of the a ones actually generalized worse like Neil shmer from C had a really interesting table where the E5

Mist troll was worse than E5 large despite being seven times as big so probably just stick to like the small ones between like 100 and at most a billion parameters but that would be my only advice that try all the good ones like GT BG five Chris Levy is asking this question about elastic search which I also get quite a lot so he asks anyone here have experience building rag application with just keyword bm25 is a

retriever at work makes use of elastic search and he said this all over the tech stack like people are already using elastic search is there basically he's asking is there a way to keep using elastic search with rag that you know about or that You' have encountered or do you mainly use like Vector database like Lance DB and things like that we tried seeing people using elastic search and trying to bootstrap off for that yeah I've used elastic

search a bit and it's perfectly possible you do lose obviously the semantic search aspect although I think now elastic search has a vector DB offering so you could add vectors to it you could always plug in you could always just do bm25 and then plug in a ranker at the end that's often if you read papers on like cross encoders generally the way they evaluate them is actually doing just that like do bm25 to retrieve 50 to

100 documents and then rank them using the ranker which if you can afford to just set up your reranking pipeline or call the C API is a really good way to go about it because you don't need to Ed your whole documents to sample how good it would be with deep learning because there are domains where you do not

need deep learning bm25 is still good enough in some bits and you know like I think it's

become very apparent like bm25 has never told anyone they would eat three rocks a day well I'm Bing half so Dimitri is asking is it worthwhile to weigh the 25 bm25 similarity score during the r reranking step as well probably not you generally just want to use bm25 to retrieve candidates but you don't need to give those scores to the embed to your cross en

coder there's a question I'm going to change it slightly so someone asks about retrieving from many documents rather than finding the best one and maybe The Tweak there is if you have a theory that information within any single document is so correlated that you actually want to try and get some diversity are you familiar with or have you used approaches where you explicitly try and in some loss function somewhere

encourage diversity and encourage pulling from many documents rather than from one I have not done that myself I know that there's different like loss methods to optimize like for diversity versus player accuracy but I don't think I would be able to give you a clear answer without something really Confidant about something I don't know much about have you used hierarchical rag any

thoughts on it I have not and I don't think it's very needed for like the Cur pipelines I think there's a lot other F steps you can improve since I think we have several answer AI people here I don't know if this is a question or request I'm eager to learn if answer AI will come with up with any books on llm applications in the future I don't think so but never

say never J me if you want to chime in yeah cuz I can't make any promises because my boss is watching so you see anything else to Ben did you say that you can't see the questions yeah they're all blank for me I saw one

earlier but they really show up spoly not sure what's happened with her and I think people also cannot up vote these so couple of cirks today you see any others here HL that you think we should pull in no not necessarily I think like probably going to the Discord is pretty good now yep tons of activity there as well I mean there's infinite number of questions so we we we can I can't I can't keep going you like okay lauran is

asking what's the best strategy when chunks when the documents when chunks or documents don't fit into the context window do you do rag in a map produce style summarize aggressively what are the techniques you've seen work most effectively so that's I think a very broad question because it's like why do they not fit is it because like every document is really long is it because you need a lot of different documents or etc etc so and how also another

important aspect is what's the like and see tolerance cuz quite a lot of the time you can make rag infinitely better but users won't stay waiting like 20 seconds for an answer so you need to figure out like how much time do I have one way that you can often see what I've done in production actually is retrieve the full documents but have another database that Maps every document to each sumar so you will have like done

your LM summarization have the previous step you will retrieve the relevant Chucks and then you will pass the relevant smise to the context window but that kind of depends on like your actual setting yeah I have another call at 10 which is in five minutes for me so if you've got like another final

question and really enjoy this presentation thank you yeah this was this was really great everything is super clear and well presented so thanks so much thank you cheers thank you cheers bye y thanks everyone