Instrumenting & Evaluating LLMs

153 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=SNBGD677_U0
https://www.youtube.com/watch?v=SnbGD677_u0

SUMMARY

The discussion centers on the evaluation of large language models (LLMs) and the methodologies for assessing their performance, particularly in the context of model evaluation types, trade-offs, and practical applications. Key speakers share insights on unit testing, the importance of user feedback, and the iterative process of refining evaluation criteria.

- *Evaluation types include unit tests, LLM as a judge, and human evaluations, each with distinct advantages and limitations.*
- *Rapid iteration and feedback loops are essential for improving AI systems, emphasizing the need for effective evaluation workflows.*
- *The importance of examining data and understanding failure modes is highlighted as a critical aspect of model evaluation.*
- *Using LLMs to assist in generating evaluation criteria can streamline the process of creating assertions and improve coverage.*
- *The significance of human evaluation in conjunction with automated methods is stressed, as human judgment can provide nuanced insights that models may miss.*
- *The need for dynamic evaluation criteria that evolve with the model's performance and user feedback is emphasized.*
- *Practical examples illustrate the challenges of ensuring factual accuracy and relevance in model outputs, particularly in applications like retrieval-augmented generation (RAG).*

Overall, the session underscores the complexity of evaluating LLMs and the necessity for a thoughtful, iterative approach to developing effective evaluation strategies.

our plan is Hamill and I are going to talk about evalue evaluation types and the trade-offs between different types of evaluation then as I mentioned in the warm-up we've got Harrison joining to talk about do a deep on Langs Smith Brian giving a case study from his work at hex Eugene's going to talk about some specific metrics that are used in llm evaluation and then shre is going to talk more broadly about evals user user experience and

workflows so I'm personally excited to see all of this before we get started we've emailed everyone about the compute credits I think in general people really want to redeem these compute credits and yet last I looked fewer than half of people in the course have responded to these emails it many of you are comfortable doing things at the last one I got to tell you it makes me nervous to see something

that you really want that I'm not sure people are going to remember to fill out the forms deadline is end of day on May 30th Pacific time I would highly highly suggest just filling out the form now we will do what little we

can to get compute credits to people who don't fill out the form but actually it is what little we can for instance open AI requires information about your account ID and so we just can't give you open AI

credits if you don't fill out the form so it's in your email we have put it in Discord please fill out the form so we can give you your compute credits and with that bookkeeping behind us let's talk a little bit about the topic of the day which is model evaluation for llm models I'm gonna hand it off to Hamil for a moment I think I was on mute sorry so what

we're going to talk about today is really I think this is the most important lesson of the entire series of workshops on fine tuning and the reason that is is if you think about a workflow like creating a data flywheel for things like fine tuning but even even not fine tuning even if you're just trying to improve your AI and iteratively make it better you need to have an iteration like a very you need to iterate very fast so the more

experiments you can do the faster you can get feedback and the more things you can try and those things can be prompt engineering can be fine-tuning can be whatever you know you need to get feedback very fast and so at the heart of that at the heart of this like iteration cycle is evals doing you know looking at data looking at lots of data and doing evaluations and it really is like when we talk about applied AI this is really the what I

would say the applied part of the AI is the evals in looking at data I talk a lot about this diagram in in a lot of detail in this blog post that's linked here so I highly encourage everyone to take a look at that give it back to Dan great so we've talked about evals at a very high level I'm going to throughout our time together today break it into three categories so those are unit tests

there are other names for them in early versions of this slide I called it assertions but these are just code you can run they run typically relatively quickly and they validate something that you expect about the responses from a large language model below that you'll see LM as a judge this is we you we've got the LM that responds to the task at hand and we hope that that's good but sometimes it'll be good sometimes it'll

be bad and then we send that response to another llm that says yes this is a good response this is a bad response that's LM as a judge and then the last is person just looks at the output of a model and says yes this seems pretty good or no this seems pretty bad so we're going to talk about all three of those today and for the sake of concreteness we're going to talk about them and especially where they are

useful in two different settings so one is writing queries this is a project that Hamil has worked on actually worked on arguably two different use cases that fall under the writing queries framework and then I'm going to talk about a project which was I started probably nine months ago which was debiasing text since you haven't seen anything about debiasing text I'm gonna give you the 20 second intro to it so

this was a project that we did for an academic publisher and they want to remove certain types of subconscious biases or stereotypes from people's writing so you could have for instance in a history paper or some social P

science paper the author of a journal article wrote Norway's mining economy flourish During the period due to Norwegian natural hardiness and we don't want to talk like it's a fact Norway that Norway mining economy may have first during a period but this

stereotype of Norwegian natural hardiness we want to remove that the company that I did this for actually has a large team of people who for a long time have been reviewing manuscripts and making these edits manually so we wanted to see if we can automate some of that with llms the fact that they had a large team doing it manually when we come back that'll be actually important when we think about the right way to scope an

evaluation for a given situation but coming back to our set of tests the first one we said was unit tests to zoom out for just a moment it we've gotten some feedback on this course I think the feedback like most people are enjoying it but the number one piece of the number one request or piece of feedback we've gotten is that people would like for us to spend more time going through code and maybe even going through code in a reasonable

amount of detail that takes time and I'm happy to spend the time but we have such little time in the workshops that what I'm going to do is I'm going to go through some code put it in a video and then put that on the course page so that I can spend however long it takes to go through code in a a pre granular level you can watch the video and then that doesn't take take away from our

Limited time as a group and so we can stay sort of high level so despite that I'm going to Ori use this code only at the very very highest level knowing that I'm going to go through it in much more detail later on I don't know what fraction of you and actually mayble to do a poll in Discord I don't know what fraction of you have used py test before this code uses py test but we could use

it without py test just plain python here we I'm just showing you like what does a unit test mean so this is a way of asserting something about the output of a model here we have this top function this llm pipeline creates the pipeline that we want to test I have a utility function that runs the pipeline and then you can see the bottom line there is it it just asserts that some expected text is in the result

and then using that I can run some specific tests like if I run the llm pipeline and give it who is the CEO of Google if the answer doesn't have Sundar Pai then the model is doing something wrong another very trivial example is this bottom one if I give it what is 2 plus three and the answer doesn't even the string answer doesn't have the character five in it then something has gone horribly wrong though actually it

could spell out the the word five so these are a common thing to H to include I think most projects should probably have some unit tests but for many things that we ask a model to do these are pretty limited and if and if you go back to my debiasing text there are many ways to rewrite something and so these sort of programmatic either contains or Ric bace tests yeah have real limitations let me h i want you

to talk a little bit more about how you think of these unit test and what you use them for yeah so I think about

unit tests in terms of like this is the first line line of defense when you are working on an AI system so my opinion is if you don't have really dumb failure modes like things that can trigger an assertion often times like is it's natural to think that hey like I can't write any unit tests for my AI because

it's spitting out natural language and it's kind of fuzzy and I don't really know like what what's gonna you know happen how am I supposed to write an assertion or unit test for this like you know everything requires a human or you might have that instinct and what I find is most of the time in practice and almost really every time that I have worked on a project in in practice is I always find dumb failure modes that

things that are going wrong with the large language model like with the output of large language model or something else that can be tested with code and I always find that if through looking at the data rigorously enough I can always find these failure modes I think it's very important to enumerate these fa failure modes this is an example of one this is typescript code but this is basically a you know this is example of a unit test from one

of one of my clients where they're just trying to check for the presence of a unique user ID that's accidentally being spilled from the system prompt into the final message and we don't want that so so that that is a that's a example of a unit test you want to abstract the logic of this unit test so you can use it everywhere like you may not only want to encapsulate it in something like a pi test you you also

want to write these tests in a way that you can also use them during production like in your llm invocation pipeline so that you can do things like self-healing Harrison might be talking about that later on when we go through different Frameworks and then more importantly you want to log the results of these unit tests to a database or you want some systematic way of tracking them otherwise you don't know if you're making progress and that's really

important we can go to the next slide and so like how do you write these unit tests so one question is like okay how do we like you know you have some application some AI application that's working off data and one kind of simple approach that I like to use is enumerating all the different features that the AI is supposed to cover and then within each feature I have various scenarios that the large language

model is supposed to handle and then what I do is try to create test data for that scenario so in this example this is a real estate CRM company called reat that actually Harrison is also familiar with they have lots of different tools and features that the large language model is supposed to to respond to like things like finding listings like finding real estate listings so in the in the in the feature that finds listings for you

there's many there's different scenarios so like one scenario is you only find one listing that matches the user's query another and yet another scenario is you find multiple listings that match the user query and then also you find no listings that match the user query so what we do is we kind of create we break down the application by all the different tools and then all the different scenarios that can occur within those tools and then we generate

test data for all of those and so like the next slide you know basically one way to think about it is either either you have user data that you can bootstrap off of but sometimes you don't so one way to one way you can go about this is to generate synthetically generate lots of inputs to the system so this is again the reack case you know you can generate lots of synthetic data just by using a large

language model this is an example of a prompt is basically saying hey write an instruction that a real estate agent can give to assistance to create CMAs CMAs is a comparative market analysis don't have to worry about what that is don't get too hung up on the exact prompt here basically the idea is you can use large language models to systematically generate pest data for all these different scenarios in features that you want your large language model

to respond to hopefully your use case is not as broad as this rehat one usually the ones I work on are not but this is kind of like a like one way to think about it going on to the next slide so I mentioned logging results to a database so when you're first starting off it's useful to to try to use what you have you don't necessarily need to buy stuff although the tools are getting really good now and we'll show

you some of them and actually Harrison will show you one that I quite like but you can get started by using existing tools so like for example I have one client this the same client reat they have metabase that they use to log all their different experiment results and then you know we started off by logging our test results these like assertions to that to see that to see if we were making progress you know across different iterations so what

you see here this is a bar chart that basically shows different error rates for different scenarios it's not important to read the bar chart honestly but you know we started off by running this in something very similar to py test and this kind of a print out of that on the lower right hand side where we have different scenarios like different tools and different scenarios within those tools and then the failure rate with of those different

scenarios so I just want to there's many different ways to do this or structure these tests I don't want you don't overfit on what I'm telling you I'm just giving you some mental model of some way you can approach it and also don't get too hung up on the tools like there's lots of different tools out there and we'll try to show you as many of them as possible through this conference I'm going to give it back to

or actually I can keep going next there I was going to say there's sort of two things that as we as you talk that jump out at me so one is we've called it unit test here you can in some ways think of the I think there are two different ways of using these so one is like unit tests they should all pass and if not then stop the stop the pipeline and another is closer to a kaggle style leaderboard

and say I don't expect them all to pass but as I run through successful successive iterations I want the number of them to pass the pass to go up and that's a way of basically measuring am I making progress or when I try maybe a fine tuning off a different base model or using a different prompt like is it better or worse and that's closer to Conventional ml I don't think either of these is like better than the others

but it is interesting to just hear as you talk that's one of the things that I've used them both ways and then I think the other detail that is important for people when they're thinking about what sort of tests are relevant for them is that you have many use cases and I think probably all the projects that you've worked on following this category where like you're really building a tech product that's going to be basing a like

a general public user and for those the types of tests that ensure the data isn't like that you're not exposing uu IDs is quite important most of the projects I work on are internal so the example I gave of automatically debiasing text we actually have someone who previously was editing text manually and now they're going to use this as a starting point and so we

tend not to be as concerned about unit tests in the unit test sense like if if something is just really bad it's not that big a deal we're just showing something inefficient to our internal employee and I think that informs which of these two types of ways of thinking of tests you want of unit tests this can never fail verse we're just trying to generally move in the right direction that's a really good point and that's kind of why we have the slide

here because we noticed we had kind of like two different types of experiences and we wanted to highlight that for students like hey there's no like right way of doing things necessarily it's like it's you have to take your use case into consideration and see like how much do effort is is appropriate for your use case like for example unit test like you know in the debiasing tech text example you know perhaps spending a lot of time in unit

unit test is not going to be fruitful whereas like in the honeycomb example that we go through this course you know as the case study that we covered in previous lessons like yeah unit test is are really good for that for example yeah and I wrote honeycom on this I should said reat but for either of them you would use unit tests and for us we said like we're editing free form text and then if it's not exactly right

that's not the end of the world but editing free form text and outputting free form text we just thought it was too rigid so we ended up not using it so the second workflow we've talked about is the LM as a judge so let me hand it to haml and you can talk about how you've done that okay so the thing about LM as a judge is is very popular way to run evaluations on llm outputs

the thing that is skipped most often is not aligning or you have to make sure you align the LM as a judge to something because you have to be able to know whether you can trust the llm as a judge and so one way I do that is to iterate on the llm as a judge and measure its correlation to a human standard that I trust trust and there's a lot of ways you can do this I like to

use spreadsheets when possible for so for example in the honeycom example which you are already familiar with in this course basically what I I did is like you know over the course of a few weeks I gave my client a spreadsheet that looked like this which basically I had them critique queries as being good or bad and write down exactly why they were good or bad and then over time successively I aligned a model to the

this human standard so in the next slide you can see like a little bit of progression of that where you know over time I was able to get the llm as a judge in the human to critique in the same exact way most of the time so we could build confidence in the LM as a judge and have a principled way of of like reasoning about it so some general tips on LM as a judge one is like you

use the most powerful model you can afford often times you need you know to do reasoning that is somewhat more complicated you know this modelbased evaluation this LM as a judge is a meta problem within your larger problem so you must kind of this is kind of like a mini evaluation system of the judge itself so just keep that in mind and and also you must continue doing this like measuring the human agreement with

the LM as a judge this is not like a onetime exercise where you try to align the judge with the human you kind of have to periodically come back to this so so that those are my tips for LM as a judge I'm gonna give it to Dan who has a interesting story about Elm as judge yeah I mean for this debiasing text project we did we observe something that we really didn't like in the LM as a judge so here for for the

sake of space I call I'm using a and b as a standin to represent what was the original paragraph written by an academic author and B as what came out of the llm and we found many many cases where if you ask an llm here's the text the original text and here is the output did that reduce the use of biases and stereotypes it would say yes and then if you flip the roles it would still say yes so no matter which one

came first it would say that that was better and we just as a result said like if you think that a is better than b and also B is better than a we just don't trust you as a judge and so our my people in general really like llm as a judge and it sounds cool but whenever most of my experiences where we've used it I've been unimpressed and in practice thought it wasn't that great

so I think you probably need to make this judgment on a case-by casee basis but overall I think Hamil you're 94% agreement that's pretty good for something that you can run as code and you don't need to bother your client every time for debiasing text this lack of transitivity in terms of if a is better than b b shouldn't be better than a that made us quite skeptical and we ended up not relying on on LM as a

judge so the last one I want to talk about is human evaluation there are many many levels of human evaluation you will hear a constant theme of from HL and I throughout this conference of look at your data so it will always be at least part of your evaluation process and you saw that a moment ago of we're going to use LM as a judge somewhat heavily we need still need to get a human evaluation to see like is the LM doing a

good job you probably want to on a regular basis just update and make sure that those don't start diverging where the LM is maybe over even overfitting to what the human has judged in the past so you probably want to like keep doing that in the debiasing case because we had a team of people and they were relatively low wage people and so it wasn't that painful for us to pay to have them do this well we said that when

we run a new model in almost every experiment we would actually send it out to the humans who do this editing as their job and say does this look good or bad and so for us it was all of evaluation so for writing queries some labor was required in human evaluation but that wasn't the entirety of what Hamil used in reat for debiasing text we said it's labor intensive and because of the details of that particular project that was okay

and that was how we did nearly everything and I I think if you can afford it LM as a judge I'm sorry if you can afford it which usually can't totally human evaluation is great so how would you how would you summarize all that Hamill yeah so I mean okay if you can write tests you have a evaluation workflow you have some way of evaluating things like that you know what you can quickly do is you can

construct a workflow where you can change things about your AI pipeline such as like prompt engineering or even fine-tuning and you can get feedback really fast because you can run it through these tests you can run it through these assertions LM as a judge maybe even human eval if you have a really dialed in system for that and you can get feedback fairly quickly and so that's the whole key to this yeah that's really all I want to

say about this part and and you know we but one caveat I'll give is we've hidden some complexity like we make it sound easy like just do this everything is going to be great no there's a lot of other things that to keep in mind and Dan will kind of go into some of those things yeah we're I think gonna highlight two of them and then the second one we highlight is gonna segue really nicely into what Harrison's going

to talk about to me if you'd asked me before I started working on these projects like what is the true ideal to like if you could have humans do all of the evaluation I guess said cost was no issue so what's the ideal I'd say have humans do all the evaluation and even have some software set up that facilitates that that seems like very expensive but very reliable I'm going to give an example of where even the

thing that seems like the most reliable can still fail so one of the projects I have talked about intermittently but probably more than any other project during our session so far is this project where we take scientific images and write the alt text which is basically just a description for people who use braille readers so that they can understand the text in images so here we have an example and you see this is a l

structure diagram which is a way of conveying the structure of a of a chemical molecule you would get this as the inp put we actually put some surrounding text around it too and then the output from the model is something like leis structure diagram of a nitrogen atom single bonded to three hydrogen atoms and that's what we want the model to Output we worked on this project for a while and here the slid there's a bunch

of bars you can't read the labels on the vertical axis but you can very quickly pick out the general Trend so the first model we built is the top bar then the next one then the next one the next one and for each of these these are just different models or pipelines we send different prompts but they're just different ways of using an LM to get the output for each we have humans rate the output and the trend you'll see

is that if you go through the first four or so models we made very very steady Improvement and then it seems like when we switched from the fourth model to the fifth one there was some drop off there and then we almost caught up when I looked at this in each of these iterations like it took some time and expense when I if you look at this I think most people would say like yeah you should just stop this project the

fourth pipeline listed here is the best one and maybe I'll catch up to that but it's doesn't seem like it's very high Roi to continue so I want each of you I bet No One's Gonna Get this to pause for a moment and think about why this might be misleading like how is it that act pipel 4 might not be as good as the ones below it so let me now you had a moment to think about it let me show you

so right before we stopped the project we talked to the people who were labeling and they said yeah it seems like the new ones are probably better than the old ones and said why aren why isn't it getting a better score and then behind the scenes we actually have a a reasonably nice all this all their labeling happen in a bespoke piece of software we just changed at the back end to reuse this model that was the one

that had the best score and we saw that it got much much worse scores than it did while we were you know a couple months earlier while we were iterating and you're should asking why is that the reason and we just talked to them and they they told us this though I think we have many sorts of evidence that reinforce it is that by seeing better and better models over time even though they had a so-called fixed rubric their

standard kept going up and up so that something that they once would have rated as a 65 now they rate as a 0.5 because it just seems disappointing compared to what they have previously seen and so a if you look at this second to last Model it happens to be a lava 1.6 34b that's been fine-tuned that actually is much better than the model that we used Midway through and it's just that people's standards had in

increas and so this is all to say that things that seem really reliable still many ways that they can go wrong the way that we in practice have solved this or nothing is truly a solution but the way that we've solved it quite a bit is AB testing we again have a piece of software that people are using for raing the quality of alt text and if we want to compare two models we randomly select which model is used

to produce the alt text and then in any given week we can compare scores and that will control for changes in their judgment over time nwor great works great for this project it's impractical for most early stage projects because you don't have enough data you don't have enough human labelers to constantly be doing the AB testing re lot lots of reasons that it's just impractical for early stage projects and those AB testing like is it the right thing for you like most

things it depends but that's I think one of the just an example of a foot gun you need to watch out for let me hand it actually back to haml and I think you should talk about right before we hand it to Harrison talk about the other piece of complexity that we've hidden and you're muted sorry the other piece of complexity is looking at your data this inside I mentioned this is the most

important lesson probably in the set of workshops this is the most important point in this Workshop is look at your data nobody looks at their data even people that tell me they're looking at their data they're not looking at their data even fellow data scientists are not looking at their data as much as they should be and kind of so let's start with what the data usually looks like so it's important to know what this terminology is so first let's talk

about what a trace is so Trace is refers to a sequence of events in engineering parament the concept of a trace has been along for a really long around for a really long time how you know and it's like basically a way that people use to log sequences of events that you like may have on websites things like that like you know as you log into website and you end up checking out in a cart things like that but

it's also it's been prevalent in Engineering Systems for a while but really for a large language models a trace is relevant because a lot of times we have have sequences of events with large language models as well so we have multi-turn conversations where you have back and forth chats with your large language model you might have rag you might have function calls there's lots of different things that can happen and essentially like these are that's what you'll re see in a lot of tools and

Telemetry you'll see this term trace and that's what it means but it's really important it's one of the most important assets you have for things like debugging and fine-tuning a lot of times you can represent these data as Json L you can be represented in many different ways but you'll see them represent as Json L and in many cases and so one thing that I like to do so one thing that's really important is to remove all friction from looking at your

data it's really nice to tell people to look at data but if you make if it's painful to look at data then no one's going to do it even you won't do it and so what I mean by that is like if you just open your traces in a spreadsheet or just a text editor and you're trying to look at it and it's not you're not able to find the information that you want you're not able to filter through

and navigate that then you're not going to end up looking at your data and it's like the most important thing that it's it's probably one of the most important things you can build and so this is a screenshot from an application like a very simple one that I built at reat that allows me to navigate the traces the data and the thing I highlighted in red here they're just very domain specific things like what tool What

scenario am I looking at is this a trace synthetically generated or human generated how many you know how many of these categories and scenarios have a reviewed so far and then some links to like you know various databases and also Langs Smith which they use for logging things like that is very domain specific it's rendered and there's other domain specific things that won't get into here but BAS basically the idea is like remove all friction and you can

build your own tools like this if you want you can use things like shiny gradio I mean shiny for python so there's a shiny for python which I used here but also things like streamlet so on and so forth the tools are getting a lot better so you know you may not have to build something specific to to yourself there's a lot of off-the-shelf tools

but what I would say is you need to make a decision whether

or not enough friction is being removed and you're seeing all the information you need I don't know if you're showing a slide with the tools or are you sh that was the sorry was the next next page sorry about that this is the screenshot of the tool with the with the domain specific things that that I highlighted and this is the shiny for python kind of application it takes like a it took me like maybe less than a

day to build very easy it's very easy to build things like this me go to the next slide don't mind and so there's a lot of ways to log traces like I showed you how to render the trace there's a lot of tools and there have evolved quite significantly since like even I worked on this project and there's a whole host of things there's things like Langs Smith which is pictured here that's a way to render your Trace Langs

Smith has also other tools and things like for writing test and doing all these other things that we have talked about Harrison is going to walk through that right after this this other tool to that you can also check out like things like identic log fire which is like a logging framework there's Brain Trust there's weights and bises weave and there's also open source tools like open elemetry and instruct about the the next slide I just want to point

out for the instruct we have JJ aair who's going to walk through in very detailed code and do a workshop on on the honeycomb problem which I've been using as a case study in this course we've worked with him to bring that example into his open source Library instruct and he's going to walk through how you would do kind of like an endtoend eval using his Library so I highly recommend that I would almost say like treat it as a required part of the

course because it's it's going to be it's going to be really good and that's that's going to be tomorrow at 1 to 2 p.m Pacific okay next slide and so so there's a lot of things that we have talked about here writing unit tests logging traces doing evals looking at data all this stuff now some of this stuff maybe you want to build a tool but honestly it's a lot of things you know to think about and you know

especially for things like logging traces you know I don't recommend building tools for that yourself like just use use there's like good tools out there use tools where you can like offthe shelf tools we will be going through so if you go back to the previous slide for a second sorry I just want to point out that Langs Smith Harrison is going to do office hours and be talking about it right here in this lesson Brain Trust will we will have

a session we might end up having a session with was and bies and then also we're going to have something for instruct so you're going to get a lot of exposure to these tools so so yeah it's best to use a tool if you can to offload a lot of these things so you can focus on looking at data so that's a good segue into Harrison Chase who's going to be talking about Lang Smith for logging in

tests in other things which I may not realize that Lang Smith does so I'll hand it off to Harrison and just format wise we should set a time aside some time for Harrison to answer questions in the Q&A right after he finish the

speaking rather than bundle all the Q&A for the end okay sounds good thanks for thanks for having me guys I'm excited to be here the title of my talk I don't really have one but the title of my talk is why HL

and Dan are right and you should listen to everything that they say because I think I very much agree with all the points that were made earlier and we we've had the pleasure of working pretty closely with HL on a few different projects and so I'll I'll I want to spend the next 10 or so minutes showing off Ling Smith but more than anything I really want to show how you can do exactly some of the workflows

that Hama had described because a lot of the things that we added were through conversations with him and through ReChat in particular around different workflows that that thought would be nice and there's some that we haven't added yet and I can I can hint out what those are but let me go ahead and share my screen and hopefully you all can see this okay so this is Lang Smith thanks for confirming this is Lang Smith this is our this is our platform

for logging and testing of LM applications it works with and without linkchain and everyone here in the class should get credits for it and we'll be doing an office hours on it as well the first thing I want to show about is is really looking at your data I think this is many people's first entry point into Ling Smith if you're using link chain it integrates with one or two environment variables if you're not using link chain you can we have we

have a few different entry points you can set a decorator on your functions you can log spans directly and what happens when you log them is you can log them to a project so I'm going to click into chat link chain here this is a chat bot over our documentation and I can see a log of all the things that were asked I can click in to any one of these things and I can see exactly what's

going on under the hood so here I can see that that I first made a call to Google llm and I got it to basically rephrase a question I then passed that question to a retriever and I got back a list of documents and then I also and then I made a final call to Google's llm and I got it to generate an answer and I can see here when I'm clicking into them I can see that

everything's rendered really nicely so we spent a bunch of time trying to make this look as nice as possible we strongly believe that people still should be looking through their and so we want to make that experience as enjoyable as possible and so you can see kind of like the system message the human and AI messages the output the documents kind of like render nicely one of the fun things that we also added was you can basically go directly

from this Trace into a playground so if you want to tweak The Prompt at all or or do any modifications you you can jump directly to there and we found that really helpful for this kind of like iteration speed all right so so that's one thing that H mentioned looked at your data another thing he mentioned when he was talking about that and I was taking notes this the past 10 or 15 minutes so I'm gonna I'm gonna referring

to my notes throughout all of this is the ability to kind of like filter and dissect data and we've invested a lot of time into really good kind of like filtering of these runs so I can add a filter I can I can filter to errors I can filter based on latency so I can get runs that took a long time for status I I tag these with various things so I'm using four different llms actually

actually five different llms and I can and I tag them and I can then filter into ones that are using openai or Google or anything like that and I can filter on feedback as well and and so this is one of the main things that we see people want to filter on because it easily draws your eyes to things that the user said did poorly or or did well and and I can also view aggregate statistics of this over

time and then I I think Dan mentioned AB testing a little bit so we actually do a version of that with chat Lang chain and what you can do is you can group your statistics by metadata so I can see kind of like various so so here we tracked various stats and there's a really cool one to look at which is latency and so I can see the latency of these different models over time and so I can see

that we have fireworks and I think open AI are generally the fastest and then and then we have Kar anthropic and Google up all right what's next on my list of notes I think after that I jump to a bunch of things around data sets and testing so as I mentioned there's kind of like two core components of Lang Smith one is the observability side which I just showed off and then the other is around data sets and testing so

first step is to actually create these data sets we support a few different ways of doing that one you can upload examples manually so let me go let me go here because I think this is a good one you can upload examples manually you can click in you can see exactly you can see exactly what they are you can modify them you can also import them from traces so if we go back to our project this is one

workflow that we see being done a bunch is maybe I want to filter to things that were given a score of zero so these are bad examples I then maybe want to click into this and I want to add this to my data set so that I can easily stop it from happening again if this is happened badad and so then I can say yes these are the same and then add it to a data set and so this

workflow of traces to data set is I think a a a really nice workflow and a good reason to have kind of like a tool that unifies your data one place anyways back to data sets the thing that I like about this example is you can see that we actually have two different splits here so one thing H said was think about what kind of like situations your application could mess up in and build kind of like data sets

that that test out those and so this is one feature we've added recently where you can organize your data set in different splits and then you can also test it on these different splits to see how they perform and so this is really useful where if you notice that there's one particular failure mode for your application you can drill in and just test that split a bunch of times rather than testing the overall thing for the for tracking the

things over time I'm going to jump to an example where I have more runs over time here you can see that so basically what you can do once you have these examples is you can kick off runs so these are normally picked off client side which has the benefit of basically you know it it work it's a very Lang Smith is very much kind of like a code first platform and so you can see that you basically Define the

function you want to evaluate and this is using the link chain chain but again this can be anything you then Define the data set that you want to run over here I just Define the name or the uid of fling Smith data set and then I def find a bunch of evaluators and so we have a few offthe shelf evaluators so here we're using this Chain of Thought llm as a judge evaluator and I'll talk a

little bit more about LM as a judge because that was brought up a bit by both Dan and HL but you can also Define kind of like arbitrary functions to run here and then just pass them in yeah you can use this oh nice we have some examples without one chain as well once you run these examples they show up as experiments and so you can track their results over time you can make sure that there's no massive

progressions like there was here and then another thing that we added that I really like to the spirit of of looking at your data is a really easy way to compare kind of like two experiments so if you want to drill into like what exactly one model got better or worse at you can see here that we highlight two two cases where this this model performed better than the other one I can easily filter in to see

what those two cases are if I had other metrics I could switch I could switch it to that I can also compare more than two experiments so I could I could jump three in here and view them side by side as well and of course I can open it up and I can look at it in a little bit more detail so the spirit of looking at your things or looking at your data input output result one result

two a few last things I want to highlight on the topic of LM as a judge we also support adding L as the judge is in the UI so that they'll automatically run every time an experiment is uploaded and so this is nice because you don't have to then run it client side and so what I can do here is I can then we have a bunch of off the-shelf kind of like evaluation prompts but you can also write your

own one of the cool things that we're working on at the moment Hama mentioned this idea of aligning human preferences with the LM as a judge and I saw a question in the chat about how to do that in one of the theories thr out with few shot examples and so we want to make this alignment through few shot examples a really tight cycle so one of the things we're working on is basically a correction flow where you

can kind of like use an llm as a judge annotate whether it did it right or not that then gets fed back in as a f shot example and hopefully you can start to measure alignment over time the last thing I want to show out just in the spirit of looking at your data and annotating data in particular is we have a concept of annotation cues in Lang Smith and so this can be used to gather human feedback basically what

you can do is you can send data points from a project into an annotation queue where it will then be loaded in in a format where you can kind of like easily cycle through data points and label them so you can see the inputs you can see the outputs you can add it to a data set you can move it to an end you can Market as done you can you can leave feedback or you can add a add a note to it as

well if you want to collaborate I'm sure there's there's a lot of things in here that I probably didn't cover but in the spirit of just trying to Echo things that H and Dan Dan said I think these are the main things I wanted to show off and so I will I will stop there and yeah happy to take any questions or skip ahead for time management I think that was really good Harrison like I know that's a mind dizy

set of features and things but it's like actually helpful to visualize some of these things because like we talk about it in bullet points and stories sometimes helpful to see like what it looks like operationalized like for me I know when I'm learning things it's helpful so that's that's why I wanted to have someone like you like show that because it really like helps glue the intuition of like okay how does it work I think then what do you think like I

think U because of time maybe we redirect the Q Q and A for Harrison into his office hours because we have three yeah and the other the other thing that would be nice about that is I should have thought about this more ahead of time when I look at the Q&A many of those came in while you an Irish be aren't necessarily Lang chain specific Q&A so if you see any in the Q&A that you especially want

to pick off Harrison that's great otherwise I like the idea of doing it during your office hours and then we'll get questions that are like more narrowly targeted to you I'll Al you know what that sounds great to me and I I'll go through the Q&A right now and respond to them because I think I can also type out answers and that way all right sounds good all right thank you guys for having me see you yeah I think okay next we're

gonna have's next yeah Brian's next and so let me just introduce Brian a little bit if you don't mind so Brian's friend a good friend he is a brilliant machine learning engineer data scientist been working in the field for a while he has his great book on recommendation systems which is actually very relevant to llms as well and what I wanted to do is like so a lot of this eval stuff is very use case specific I wanted to bring bring another

expert in who I know does really good work and have him describe or you know you know have specifically Brian describe like his approach to doing evals and instrumentation and things like that and like walk you through his workflow to give you just yet even another perspective so that Brian hand it off to you thank you so much yeah I'm gonna talk a little bit about some of the things that you've already heard I'm gonna underscore some of those lessons

but ultimately I'm going to try to give you a sense of how I think about making this real I always click that wrong button no worries looks like it's frozen I think share means share the and so today I'm gonna talk to you about spell grounds can you hear me yeah I can hear you okay cool today I'm GNA talk about spell grounds for prodigious press the digitation so you're going to see a little bit of magic themed stuff today

because I work on a project called hex magic but spellgrounds is the name of our internal Library for developing and running evaluations it is going to be a combination of like sort of like systematic evals and use case specific evals and unit tests and regression tests all that all in one thing and we're going to talk about why but right off the bat I want to give you a very opinionated position on what evals

are about evals serve one or more of the following three purposes they help you understand when a capability is good enough to present to your customers they help you sleep well at night they give you confidence that your systems not behaving badly or they help you debug later when things go Ary I am pretty convicted that these are what evals are for and what they're about when you think to yourself about

the things that you've learned so far in the course and the lessons that you've heard about goody and how you can use evals they usually in my opinion always fall into these free buckets you'll also see this in the market when you talk about evals tools so I'm G to talk about these three lessons as part of the things I cover so first up I'm gonna tell you a little bit about some things you should avoid we're

going to call this miscasts and fizzled spells these are mistakes I've already made or I've narrowly avoided the first one that I see people make a lot is they think llm valuations are entirely a new field they're not they've been around we've been doing it for a long time and the experts are data scientists we've been mapping complicated user problems to nuanced objective functions for over a decade some of us personally for over a decade and the field for close to 20

years we've got really good instincts here on what it means to measure unpredictable outputs you may think that the nuance and the beauty of your llm outputs or some ineffable thing they're not I promise you I believe that you can coers whatever it is you're trying to get generative AI to do for you into reasonable and quantifiable performance so let me give you some

examples for code generation you should be considering things called execution evaluation run the code and see if it does what you expect in my workflow I need to evaluate SQL and Python and R I run the code that's generated by the model I compare it to the code that I've run in the Target setup and I make sure that the outputs have the same state that's how you evaluate code generation right now everybody's really

excited about agents what does it look like to evaluate agents well one important step is planning you you should be thinking of that as a binary classification data set the steps in the plans you should think about which one are required steps which one are steps that have some looseness turn those into binary classifications turn that entire sort of State machine into a set of binary

classifications you may think okay Brian well those are easy what about something complicated like summarization you should be checking for retrieval accuracy Does it include anything that references the important points that your summarizations in your target code or your target response include so this is probably the biggest I would say trap that people fall into is they think that they can't make their evaluations like old school data science

evaluations here's an example of how I compare the output of data science code very simple all it is is massaging the data frames and saying is there any way possible that these data frames contain the same data this is a blocker that I hear a lot of people Express and actually during my interview Loop people tell me that they can't think of how to evaluate data science code other than checking if the cod's the same well if I

ask you how many customers do we have this month well the response from the agent may be a different data frame shape but as long as it has that one number in there somewhere then that's good enough for me this is the kind of code that you write these are called relaxations okay next up people fail to include use case experts in eval creation your users or experts on the use case of your llm application they

will understand what good looks like if you were thinking about rolling out an llm application without talking to experts about what the final State should look like I think you're goofy go talk to them understand what they want while we were building magic charts which is our sort of like llm generated data visualizations in our platform we talked to the data team what are some example prompts to go from existing notebooks into the target chart the perfect version of the chart we

worked with them to build out a set of them it looked like this The Prompt create a bar chart to show the average number of passengers per month using the flight sta set blah blah blah and this is the chart that they wanted to see so what do I check in the output of the llm do I check that it looks like this pixel to pixel hell no do I check that it's selected the right chart type yes did I check that the time

axis is a date time axis for X yes do I make sure that it's doing an aggregate function on the y- axis yes noticing that these are all starting to look like binary evaluations next up people wait too long to make evaluations it should be part of your Dev cycle this should literally be part of the RFC creation here is a literal snippet of the RFC for a recent project

where we're working on sort of a new editing model this is in the RFC it's required to be part of the RFC now and you're required during the dev cycle whether you're a fullsack engineer an AI engineer or a product manager on the team you're required to be writing and thinking about evals this early in the process there's no excuse to get something close to prod and then say well I should write some evals I think people fail to recognize

product metrics and evaluation metrics are similar but they're different you you'll notice this look at your data I promise I wrote that before today's lecture from HL and Dan we are all in alignment that you have to look at your data but you shouldn't mistake your product metrics for your evals they are distinct product metrics will give you a lot of intuition about great evals these production logs aren't sufficient for building evals we saw some examples of digging into production

logs to give us some intuition for how things are behaving that's really important but that can't tell me all I need to know for making great evals what it can tell me is sort of how to build the next set of evals I don't personally have access to my customer data I can't go into their data warehouse and query it so it's actually not possible for me to build evals on their warehouse that's a good thing but what I can understand is the

kind of questions they ask and then I can go build data sets and Custom environments that I think are representative of those use cases so this is the push and pull of product metrics and evaluation metrics buying an evaluation framework doesn't make it easy now this might feel like a little bit of counterprogramming this is not intended to be a criticism of any specific eval framework but what I can tell you is I don't have an eval

framework to sell you and so let me tell you what I feel about this topic the hard parts of the evals is not the library I built a simple framework on top of unit tests in a few weeks it lasted nine months we ultimately had to rewrite it but that was also one Sprint the best eval product is a Jupiter notebook it doesn't have to be a hex Jupiter notebook just a Jupiter notebook interact with your data get your hands

on the damn data be able to slice and dice evals are hard because they they require you to understand the user stories and the diversity user inputs sorry and finally eval companies they haven't put the llm applications into production remember the saying don't ask a shoe seller what's going to help you dunk ask Michael Jordan I'm here to tell you I've put this stuff into production I promise you you do not need

an evaluation framework until you start feeling the pain then it's time to think about it this is what my evals framework looks like one class this is me setting up the idea of assertions how do you evaluate in a very flexible and relaxed way whether the eval name space and the target name space I.E what the agent responds with and what happens after you evaluate it

and your target look like this is all the rapper code please please please invest in the things that matter don't invest in complicated Integrations early reaching too early for llm assisted evaluation you're going to hear a lot of people tell you about llm as a judge there are no free lunches I think llm judging is extremely valuable but it is not a free lunch what it's going to give you is directional metrics things to look into it's going

to start giving you some hints once you built some real evals that actually like kind of give you important signal llm judge can help you scale llm judge can also help you look at production events to kind of turn those into great evals at scale but you have to do this methodically do side bys side evaluation on new treatments use multiple judges multiple models multiple

shots check for human alignment randomly and periodically the best paper I've seen the best like writing I've seen on this topic is by Shrea Shankar check it out this is what it looks like to do llm judging systematically over time this is we made a relatively minor change in our context and we ran seven experiments here and we wanted to see the performance on different versions of the model and each time we're using the judge to tell us what's better old or

new old or new and we're doing this over 2,000 EV so this is what it looks like to try to use llm judge as a tool part two moderating magic how do you build an eval system yourself magic is an AI co-pilot for data science that lives in hex it can generate SQL that's specific and knowledgeable about your data it can string cells together using different kinds of cells to write polylock code chains SQL python R and Native charts it

reacts to code edits the user does or ask the user or allows the user to ask for edits and fix it specifically First Step ragy vals you should evaluate your rag like a retrieval system rag is retrieval treat it like such if you start thinking about chunking and indices and multi-index and hybrid and no take a step back label some data take the queries from your evales and produce the best documents for those queries and measure the hit

rates that's all you need to do to measure rag don't get too excited about your rag system unless you have a clear Baseline your retrieval scores whether the semantic or lexical aren't calibrated either so don't treat them as confidence estimates just a little tip I've gotten burned by this this is what it looks like to evaluate rag on the right hand side we're looking at different rankers on the left hand side these are all the embedding models we try for one

particular workflow don't be shy try a lot of things look at the variance in performance over there that should tell you that this is important planning evals for agent-based systems you have to evaluate your planning if you're using a state machine treat it like a classifier check its choice at every step if you're asking planning to include downstreet prompt generation the quality of those downstreet prompts are probably they were for us at least it took me a

non-trivial effort to get the planning prompts to be anywhere close to what the human prompts look like don't forget to evaluate that agent specific evals this is the easiest kind of eval you have an agent it does a specific thing good test it think about structured output how often can you get the agents to respond with structured output and if you do how can you tie the agent relationships together with tightly spect out API interfaces

and then evaluate that consistent consistency final stage EV vals a lot of agent chains need a wrap up or a summary don't forget to evow this too sometimes you get kind of stupid you'll get the summary talking about things that the agent never did that's terrible as a user experience trust me I have accidentally done this too much context about the agent chain can make these quite bad don't serve everything from the entire workflow to the final stage

eval it will be noisy and frankly the tokens get kind of crazy finally experiments in llm generation are repeated measure designs they're not AB tests when you make updates and changes and Bug fixes to your agent's workflow treat it like an experiment and measure it thusly doing betters on your EV vals is is a good check but you have to for significance and don't be afraid to rerun production events through the new treatment so

historical production events through the new treatment and use automated evals to compare you're logging your production events aren't you you didn't forget to do that did you this is what it looks like to run experiments on the left- hand side this is repeated versions variants of a different approach and you can see the error rate is going down on historical events on the right hand side that's LM as a judge don't sleep on

this and then I have one bonus lesson for you production end points minimize drift I thought I've got this production system I'm going to build a clone of the production system in my evals framework so that I can keep things as similar as possible to production and some of you are already shaking your head Brian you're an idiot

that's never going to work of course it didn't work we had to refactor our eval framework because we

built a clone tightly coupled systems that aren't actually identical this is standard software engineering like don't do it and I did it and I regret it don't be like me make your eval framework directly connect to your production environment make them end points call those endpoints use that as the same backbone that calls every single thing in your evales framework but make sure that every step

is exposed be able to hook in halfway through that workflow make modifications upstream and see what happens in the end that's how you keep these tightly coupled in a sane way that's what I have for you today you can find us at xx. you can find me on Twitter at be Bishoff or LinkedIn Brian bishof I'll pause there for any commentary but I think we're on a tight schedule today so okay I'll ask you one

question from the audience of course how are you organizing all your unit test where are you running them local GitHub actions Etc and where are you logging them to in practice Yeah so we are very fortunate that hex notebooks can be scheduled and so I run Jupiter notebooks that are scheduled so believe it or not I orchestrate and run all of my evals in Jupiter notebooks no no asterisk there that means that a they are incredibly

reproducible B if I go back to an individual experiment and I say huh that performance is very different than I expect I can pull the individual eval logs and literally look at every single response from the agent that's an amount of power that no other system is going to afford now granted could set this up to log to your data warehouse and you could use a downstream sort of notebook to like evaluate all this but frankly like I'm very fortunate that I

just have access to hex like this so I just do it in HEX that's really fascinating that should be a whole podcast in itself about books and production could go and like would love to probably talk about that happy to okay no that's good we have a lot of questions but I think for time management sake I guess like we can what do you think Dan should we do one more or should we move on yeah let's do

one more okay okay so Alex strick asks for unit test on their own take a very short time to run but lots of these individual tests together take a really long time do you have any tips on batching these together is there a good framework for running through a bunch of these queries in parallel yeah and he says I have a bunch of these unit tests in my code base but it still take a few minutes to run through

sequentially yeah so we have a batch version of The evals that runs on a schedule and I don't like you know it runs on a Friday and I like collect my results later on but actually again like I'll hearken back to the good old days of data science don't be afraid to do two things one look at the sample that is most concerning to you and two don't be afraid to use bootstrap sampling like bootstrap sampling gives

you a really good picture of the overall like behavior of the population from a couple random samples do that that's a great way to get early signal on these things but also let's be real some of my evals almost never fail some

of my evals fail every time and that's a good thing if your eval Suite is 100% passing your eval Suite is not hard enough make it harder you should be really targeting evals that succeed 60 to 70% of the time

because otherwise you're missing out on the signal of improvement how will you even know if you've made anything better and so the way I tend to look at this is I tend to look at the marginal ones which ones flip when I change things if I can't get them to change actually to be honest I'm eventually gonna delete them like they're not doing much if they're never failing if we've moved on in the world from gbd 35 like quality on

a certain test well then why do I need to test it every time that's sense as scientist like to say there's no variance there's no signal okay that's great that's a very great presentation Brian really appreciate that I think people are commenting they really love this presentation by the way awesome awesome glad to hear it so the next guest that we're GNA have is Eugene Yan he's also a good friend who I've known for sometime now he has he is

quite prolific and writes a lot about machine learning large language models you name it the thing that he will be and sorry Eugene is a senior machine learning scientist I hope I'm not getting the title wrong at Amazon I'll let him correct that but what is going to be talking today about is metrics so a lot of people are asking like what metrics should you use we're talking about like okay measuring writing evals having metrics but they

haven't like really gone into metrics so Eugene is actually going to talk about that in more detail Eugene thank you that's way too kind all right everyone we are almost at a one half hour mark and I will be going fast and there's going to be a lot of code and in fact it's just all code and graphs so I hope youall pay attention I'm dropping the notebooks that I will be sharing in the slack in the

Discord right now so don't bother to take notes don't bother do screenshots all this is fully available so all I have right now is six cells Mo of slides and three notebooks all right let's go so the question I have is how do we evaluate some summaries on factual inconsistency or hallucination right everyone's using some kind of LM to summarize stuff but how do you know if it's actually correct well if you actually look at it where we

are human eval you'll find that hallucinations happen about 5 to 10% of the time and sometimes it's pretty bad so the way we can do this is with an evaluate the model right the input is a source document and the summary and optionally the label if you're fineing on it and the output is the probability of the summary being factually inconsistent so we can frame this as a natural language inference task so natural language inference is a

classical NLP task where by given some premise and hypothesis we have the label of it being so imagine the premise is John likes all fruits the hypothesis that John likes apples is entailment and the hypo this is that John dislikes apples says contradiction and of course there we also have a neutral which is we can't have we don't have enough information to tell what it's correct not now if we apply natural language inference to factual inconsistency

detection what happens is that we can use contradiction the contradiction label as Vector inconsistency so

imagine we have a document maybe this is some talk abstract for a previous talk I did eugin talk is about building LM systems Etc so the summary is the talk is about LMS that will be entailment and the summary of The Talk being about apples that would be contradiction so then all we have to do is to get the probability of contradiction and there

we have it we have a factual inconsistency classifier or hallucination detector evaluator model all right so my objective here is I'm going to show you how to F tune an evaluator model that can catch hallucinations on the factual inconsistency Benchmark this is my new KGO then we're going to eval the evaluator model through each Epoch and we're going to see how we can blend data to make this eval evaluator model way better then optionally you can use this

evaluator model to then eval generative models I know it's kind of meta but we're going to eval the evaluative model which then evales generative generative models and then you can also use this evaluative model as a guard ra right you going to summarize things in production and then you can check hey is a summ factually consistent or not we're going to see how how this happens so this is a three body problem we'll first examine prepare and split out data no this is

completely coincidental I have not spoken to the instructors about the importance of looking out at your data but everyone has mentioned that and I do have a notebook for that after we're going to fine tune on the factual inconsistency Benchmark and then we're going to blend in data from the unified summarization Benchmark and see how that helps okay I have a few appendices here I have three writeups on LM evals and hallucination detection and all of

domain fine tuning all which are compress into a 15 minute session for you here and I also have some slides on evals and fine tuning and how they are two sides of the same coin so next let's prepare some data so over here we have the factual inconsistency Benchmark it contains one sentence summaries from the CNN daily mail and the xam news articles we exclude the CNN daily meal data because it's pretty bad I won't show you how bad it is but you can

look at it yourself so now here's an example of the XM data the first two rows you see that they have the same input and the next two rows have the same input so how this looks like over here is that for the same input we have a choice that is inconsistent and we have a choice that is consistent so if you if you look really hard at this you will be able to understand why is it inconsistent or

consistent but if you just briefly glance through it you might actually miss it and a lot of times I've been I've been looking at this myself and I'm like wow this is a really difficult data set so here's the CNN daily mail data set and you can see it's full of it's full of XML tags and quite a number of them actually have ads in there so actually we just discard them so FIB starts with about 3600 rows of data

after we excluded the CNN daily maal data set we are still left with 3100 rows the authors themselves of this data didn't even bother using CNN too much because I think they just label 100 and they realized actually it's

not that not that good and they just invested more of their resources on XM so that's the FIB data set So eventually what we're going to happen what we're going to have is and of course we split it we split the data

we're going to group it by input this ensures that the same article doesn't appear across train and Val right so we want to only make sure that the same data only appears in either train Val or test so there data leakage So eventually what happens is we have and then of course we try to balance the data so there's only one positive summary and one negative summary eventually what happens is we have 700 data points for

training of which it's 350 positive and 350 negative and our Val sets are only 150 so that's the factual inconsistency Benchmark next let's look at the unified summarization Benchmark the unified summarization Benchmark is slightly different it's based on summaries of Wikipedia articles so you can see over here again the first two rows they are the same and the next two rows they are the same and when you look at it very carefully let's let's look at a second row here it actually points

out what the inconsistency is and it's highlighted it's surrounded in the red boxes which is the summary actually includes this additional data which is not in the source but there are times when you look at it over here I it took me quite a while to spot what the difference here is and the difference here is the word the and and even though not having the word the is not a hallucination problem it's more of a grammatical or sentence

structure problem it is part of the data set so suffice to say I didn't clean this up but it just goes to tell you that you do need to look at your data when you're fine tuning to try to understand the quality of it okay so we get the data we try to prepare it and what happens again is this we have a summary sentence where the label of zero is correct and the label of one is

incorrect and of course we do the same thing train test Val split Etc in the end this is the amount of data we get so we're going to first look at the factual inconsistency Benchmark so over here we load the data we do some we tokenize them all up front in a batch and then over here we're going to fine tune our model the model we're going to fine tune is a what we call distill bot essentially it's you

can think of it like an encoder decoder version of B from mea but it's also fine tuned on mnli which is multilingual natural language inference data and of course we have our parameters here nothing too interesting we use Laura we apply Laura on the qkv vectors out projection etc etc in the end the number of the amount of trainable parameters is less than 3% this allows us to fit this on a very nice small GPU so over here I have some custom

metrics that I track so this would be akin to the callbacks that Wing mentioned about during Exel the Exel Auto session so over here at each EO or at each eval I actually firstly I pre-process the logits so if you recall NL actually produces three probabilities what I do is I only I only care about the entailment and the contradiction probability and I do a soft Max on them which sums up the probability to one and

then I just get the probability of contradiction so essentially those I care about and then I also compute some custom metrics that I have essentially PR a r recall and precision and recall and precision I have an arbitrary threshold of 0.8 where I want the model to be pretty I want the model to be pretty confident okay so I have a custom trainer here I won't go into it right now but I'll going do it in the next in

the next notebook so again some standard training arguments nothing special here so let's and of course I also have some plotting code here again nothing special here so let's look at how the model performs before any fine tuning before any fine tuning you see the ROC a is at 0.56 if you recall an Roc a of 0.5 means that it's really a coin flip and the model is not doing anything better than charts so you can

see okay R's coin flip and what I really love is the graph all the way on the right which is the overlaps of red and greens the reds are the we have the ground truth we know what is inconsistent and the greens what we know is consistent and we can see the overlap and you can see that in this case the model just cannot distinguish the overlap it cannot distinguish between them before any fine tuning now

note that this model has already be fine-tuned on M&L but still is doing a pretty bad job so now let's start fine-tuning and so you can see these are the custom metrics I have usually if you don't have any metrics you probably only get loss training loss and training loss but I do have extra losses I do have extra metrics such as p r recall and precision so let's just focus on R you can see that R increases from 0.56

we have over here to 0.65 not too bad but at least it suggests that the model is learning but what is very disappointing though is that the recall at 0.8 does not go above 10% so in this case this is just not usable right we this model just cannot identify factual inconsistencies so we check the evals after fine tuning this is note this is on the trading set we train on this and we just check on this to make sure that

our model can learn we see R AOC is 0.71 and we start to see a little bit of separation of distribution right still see a little bit of Separation but on the validation set RC 0.66 not that good and the the separation is pretty bad and over here on the test set which we have never seen and which is we are not using to pick our checkpoint RC is slightly I I think it's not not statistically

significant but you can see the separation of distribu is pretty bad so next let's try let's see how we can f tune on a different data set we fine tune on the unified summarization Benchmark followed by the factual inconsistency Benchmark so the factual inconsistency Benchmark in this case is the data that we care about you can imagine in your own use case you would have some kind of summarization data and all you care about is how it performs on

that data so what you can do is you can take all this open source permissive use data sets with both of these are and you can use them to bootstrap your own model by blending in the data and we'll see how how how you do that here again so we have the fact so again to remind you about our data our factual inconsistency Benchmark we only have 350 we only have 700 training samples but the the unified

summarization Benchmark we have 5,000 training samples and I deliberately split a big chunk of it into the validation set right so you can see that USB has almost 10x more of the factual inconsistency Benchmark so by using some kind of external data you can improve your own models and in this case the what what I'm what I care about is hallucination detection or summarization so again we do some kind of batch tokenization so we don't have

to to tokenize on the Fly and we set up our models nothing nothing special here so what is this custom trainer this custom trainer is because at a point of me trying to do this which is about six seven months ago hugging phase trainer didn't allow for evaluation on multiple data sets so this is really just copying the trainer code and overwriting some of the overriding some of the methods that it has specifically eval and maybe lck SL

you can just use this code if you want to so how it looks like here is that without the custom trainer all you could do was this which is eval data set you provided a single eval data set but with the custom trainer what you can do now is this you can provide a dictionary of data set to the data to the eval data set and it would just work so again we have our usual we

have our usual visualization code and this is the same thing USB what we saw R 0.56 this is oh sorry that was FIB the data set we care about R is 0.56 and this one for USB RC 0.66 a little bit better but the Divergence looks a little bit funky it in the in the previous case you can see it was a little bit too strict most of the probability was about 0.75 over here it's a little bit too

lineate most of the probability was close to one so let's look at this so what we going to do is we're going to find tune this model and let's just focus on the USB metrix first which is USB P RC you can see that the USB Ro R AOC very quickly ramps up from 0.7 6 to 0.94 that's pretty freaking amazing and recall and precision is 0.75 and 0.93 that's really good I don't think in in production I don't think your data is

going to be as clean and you can achieve something like this I I you we also probably want to TW the threshold to maybe either buyas to recall Precision but let's look at the let's look at fit's R you can see fips R and then what I'm going to do is I'm going to split the screen here for a second so you can see previously fips Roc a went up to 0.65 right over here by solely fine

tuning on USB not a drop of fit data we could get the fit Roc Au to go up to 0.64 or 0.63 that's what the heck is going on man I mean it's the same task but these are completely different domains but the thing is look at the recall the recall has gone up from Z from 8% to 25% to 25% etc etc that's a huge world of difference from what we were seeing

previously right where we start be at 6% 4% so what we going to do is we we fine tune on this and we can see that it's superb on USB over here you can see that it does superbly on USB and this is the validation set you can see oh my goodness the aror cc is freaking amazing and look at the Divergence right over here you could probably just cut at 0.8 or 0.9 and you have the clear fact

consistence or over here you cut at 0.1 and you have the clear factual consistent it depends on how conservative or how linear you want to be or with your false positive for false negatives and now let's look at it on the validation set for fit the the data set that we care about we see that the ROC Au has not improved that much right so this is the previous Roc you see this is the previous one and this is the

current one it's like 0.66 versus 0.64 the separation is not that good so we may feel like hey some it didn't work here but now let's give it the same 10 epochs of FIB data which it has never seen before the same training okay so you can see the same training over here previously R was 0.65 and recall never got above 0.6 now with this you can see fit R and

we don't actually have to measure it on USB I just because I was just curious how much adding in additional data will cost us in terms of alignment tax on USB you don't actually have to track this but I did oh my God the RC imil started at 0.75 and then went up to 0.86 and look at the recall it's like 10x higher than what we have previously 0.57 and precision is 0.8 0.93 that is insane right

so all in all this are some metrics on how I would use to evaluate a classifier model so now this is here's how it looks like in the end our fit test Set so you can see the test set previously with only fit data on the left and here's the test set with fit data and USB data on the right and this model with maybe a little bit more data we can probably get get to higher recall maybe

0.8 and I think a good balance is 0.8 0.8 of 0.8 0.8 in terms of this evaluator models ability to catch hallucinations but so now you can do this to evaluate your generative model right I mean we have certain engram metrics like Rouge and meteor or you can use LM as a judge which is very expensive but this model it's super fast every query is like 20 milliseconds super fast super scalable it's fully

within your control and now what I will ask you is how could you fine tun evaluator models to help evaluate on your task or in evaluate summaries on relevance information density and okay that's all I had so now so long sry short this is my evaluator model to evaluate my summaries on factal inconsistency and eventually there will be more evaluator models to evaluate on relevance and informational density Etc so now go and find you in

your own evaluator models that's all I had that's really great Eugene I think that was like really great run through of like a lot of different things that you might want to think about with like concrete examples I really recommend everybody check out these notebooks I went through them a little bit fast but you know this is recorded you can always slow it down you can step through these notebooks also you know at Times Eugene shared some additional resources

for like his writing and that really gives a lot of color to these things I highly recommend you like check all these things out you know make just make sure you don't skip that because I I really enjoy that those writings myself let's see if there's a let's see if there's a question let me open up the Q&A

okay let's someone's asking how do you think about evaluating agents that's a great question I would just Echo what Brian said which is evaluating agents it's a step-by-step process I would break it down I mean I actually have a recent post that I just wrote over the weekend which is it's essentially about promp thing but I can talk about how we evaluate agents here so one of it is you can split the catchall prompt to multiple smaller ones so here's an

initial prompt where we try to extract some we try to summarize a transcript right so over here you can break down a transcript into and extract the list of decisions action items and owners this is just a classification metric right simple now over here the second step we ask it to check the transcript and extracted information to make sure it's factually consistent again a classification metric now the final one is given the extract given the extract

the information write it out into parentheses or bullet points now this is a informational density it's a relevance it's a writing eloquence question so I know hey Eugene do you mean to be sharing your screen oh shoot I did I did okay I did that earlier it was it's no problem it's hard so sorry guys so here's how I would imagine you have an agent again to summarize meeting transcripts that's the simplest example you could over here you could

evaluate how well the agent is extracting a list of decisions extion items and owners again this is just a classification right or extraction you can it's just precision and Recall now again another one to check the extracted information against a transcript you can imagine that this is the factual inconsistency model we talk about think step by step and check if the extracted information is actually consistent again this is a classification task now finally you are asked to rewrite the

information into bullet point summaries you can think of this as a information density task or writing style task which is a little bit more soft not as straightforward to measure their classification but maybe a reward model might work so that's how I would do it and Alpha codium had a really great post really great piece where they actually split up code generation right into multiple steps so you can see each of these steps here you could probably

evaluate each step here so that's a longwind story to say longwind answer to how I might evaluate agents the same way as I would evaluate a multi workflow that's great okay well thank you Eugene the next speaker we have let I just want to give you some background for the next speaker so a lot of times when I talk about evals as you can tell there's a lot of information about evals like how to organize them what tools you

should use metrics workflow things like that I always find with all of my clients is that people get stuck like how do you write how do you write evals like where do I even begin how do I think about it maybe and they can maybe write one one test or two tests but then they kind of have a mental block and I I think like we're still in like very nent period of all this tooling and like how

to go about thinking about evaluations Shrea who is you know who who is a prolific researcher in this area of like llm Ops and also specifically evals has done a lot of research on things like ux workflow developer tools and tools more generally she's been doing research for a really long time even prior to large language models on

machine learning tools and workflows and things like that and so I'm going

to hand it over to Sha Sha is going to walk through some of the research that she's done around you know workflows around eval and ways to think about evals that I think is really helpful for everybody so with that I'll give it over to sh up for the great intro super nice of you I think Eugene you have to stop sharing so I can share sorry to boot you off shoot I didn't know I was

all good okay so let me know if you can or can't see my screen yeah I can all good great so today I'm going to give a pretty short talk on some recent research that I've been doing with a lot of with many amazing collaborators across many different institutions and companies and the theme of The Talk is you know how can we use llms or AI to scale up our own human decision functions how do we scale up the vibe

checks that we know and trust with llms and I have a brief intro slide in case people don't know who I am or in case HL didn't intro me but I can skip this but basically I am a PhD student this talk will be a little bit more high level I study how do people write llm pipelines how do people evaluate them and how can we improve the experience for everyone and I also do ml engineering an AI startup and I like

to think about you know how can we work with data at reasonable scale how can we ensure good data quality and anything around the people in ml and llm Ops so without further Ado I will get into my talk I don't need to go over this probably I mean you've been in this workshop for so long but you know we really really like LM pipelines because their zero shot capabilities can enable intelligent pipelin lines without having

to train models so if you take a look at a bunch of prompt templates from lsmith you'll see that people are doing all sorts of stuff you know they're using llms to write code reviews they're using llms to convert YouTube transcripts to articles and when you can of write a p template or pipeline around this and I'll illustrate with this figure say we have this YouTube transcripted blog post pipeline we might feed in some input document which is a YouTube transcript

put it in some prompt template that has some instructions and then expect to get some transcript or sorry some blog post out of the entire pipeline so this all looks great but the problem is when you try to do it at scale the llm doesn't always listen to instructions so maybe you have an instruction around HTML structured and one out of 10 times it doesn't output improper HTML structure or maybe you have a sentence that says you know avoid copying

sentences directly from the transcript but somewhere in the middle of the document even GPT before might exactly verbatim output the same instruction and the theme here is you know no matter what as we suppose we fine tune these llms on our tasks there's no guarantee that it's going to listen to every single instruction that we've included in the prompt we need some form of guardrails or assertions or evals to be able to quantify you know how well does

the llm do our task and listen to what we Define as good or bad and that's where our kind of vibe checks and rules and guard rails come in and insight from traditional ml is to Simply put rules and guard rails around the model to detect bad outputs and correct them or even rerun the pipeline but this is really hard to do for llms this goes back to what HL mentioned before it's difficult to even get started thinking

about what does accuracy or good even mean for your specific task your outputs maybe 10 of us are trying to write pipelines that are converting YouTube articles or sorry YouTube transcripts to articles but maybe we all want different formats or we want the response to do site SL something slightly different right so our Vibe checks are all going to be different making this a pretty hard problem you can't just kind of use what somebody tells you off the shelf

you've got to come up with your metrics yourself metrics might be complicated requiring humans or even llms to evaluate say something like tone if we want the tone of a blog post to be informal or we want it to not sound like an AI right how do you encode that into something to evaluate and every prompt task application is different all of us are going to have different metrics even if we're trying to do the same task or different

implementations of the metrics if we're trying to do the same tasks and I like to think about these Vibe checks along or guard rails or evals or whatever you want to call it in general along the scale of you know generic to task specific on one hand we've got common MLP metrics that you know model providers talk about when they release new models which is great but doesn't really tell us how well those models are going to do for our custom tasks we've

got something in the Middle where we know of you know good metrics for common architectures for example rag pipelines we know that faithfulness is a good metric from the ragas paper so that's great but we really also want to be pushing towards these tasks specific metrics so if you have an exact structure that you want your output to follow not just Json but a specific you know you want at least two of those Json keys to follow the same pattern you

want some more finer grain constraints on that you know that goes more towards the vi checks and if I showed you one access in the previous slide from generic to task specific but there's also know another act to consider which is how simple or scalable is the method stress testing prompts in chat TBT don't really scale especially in production and then fine-tuning evaluator models are pretty high effort because you have to constantly be collecting data and determining

whether this is good or bad and then be able to find two the model Vibe checks performed by humans don't scale but we shouldn't discount them because most people do this and they're quite effective especially in the early days of prompt engineering and what we really want to do is move these Vibe checks towards the upper right quadrant and codify them you can call them validators you can call them Asser you can call them guard rails I honestly don't know

what to call them but the idea here is to have a set of task specific constraints or guidelines that you feel

confident aligns with what you think is good for your task so a lot of our recent research has been in developing evaluation assistance which are tools that Aid humans in creating these task specific evaluations and assertions that how that align with how they would grade so in my talk I'm going to briefly cover you know what do

we want to think about in how how can you build your own evaluation assistance the key idea here is to use llms to scale not replace your own judge J ments and decisions and I'll talk a little bit around different parts of the workflows and how we can use LMS to do that I'll start with how can we use llms to autogenerate criteria and various implementations of that criteria and then I'll talk about a reset mix

initiative interface that we built to develop custom insertions and some lessons that we learned from you know briefly prototyping this interface with a bunch of LM experts all right so let me jump into bootstrapping criteria okay so let's POS this concrete example here let's say we have a document summarization pipeline but the summarization pipeline is for medical documents so there are some additional examples or sorry additional instructions like return your answer and

markdown because you want it to be a report that your doctors are going to read in a custom interface and maybe you also have some instructions like don't include any sensitive information like race or gender and have a professional tone so you can see how you know these kind of get pretty specific towards the end user having assertions gives you a fine green view of correctness or goodness of llm input in output quality and every custom llm

pipeline should have thumb table like this I but the challenge really is coming up with the criteria what are the columns that you want here and then good ways to implement this criteria so some of them for example can be implemented with code some of them might not be able to you might need to use an llm to evaluate something like professional tone and Engineering that prompt itself can be hard right you're already engineering your main prompt so

engineering the validator prompts is a little bit excessive so how can we enable humans to efficiently come up with good and bad examples of professional tone to seed the validator props for example so an overview of this problem which we talk about in our Spade paper is how can we generate a small set of assertions that have good coverage with what humans think are bad outputs and also have good accuracy so challenges here are how do we find the right

assertion criteria desired by the developer and how should we guarantee the coverage of failures with a small amount of assertions right we don't want to give you thousands of assertions to run and production thousands of guard rails because monitoring that or visualizing that would be a best in our Spade system employs a two-step workflow to do this first we generate a bunch of candidate assertions with llms and then we filter them based on human preferences so our Insight here for how

do we generate custom assertion criteria is that the criteria are hidden in prompt version history so when humans are iterating and improving on the prompt we can tell what is it that they care about and what are

unique mistakes that llm makes maybe their document summarization pipeline starts out with template like this which is very common a lot of Doc summarization pipelines will start out with a prompt like this and then when trying it on their data

they might notice that sensitive information is included in the summary and the specific application developer doesn't like that so they add an instruction that says don't include the sensitive information in the summary then maybe we might see another human generated prompt Delta or prompt edit that says do not under any circumstances include sensitive information so what does this tell us this tell us that the llm is kind of bad at determining what does sensitive information mean doesn't

listen to that instruction I don't know I'm a lot of conclusions there but so forth right you can imagine looking at how humans evolve their props to determine what it is they care about in what magnitude right if you edit the same line maybe 15 times maybe that's a sign where the LM is a little bit more bad there than in other places so what we did here to build the first part of the space assertion generator was to

look at a bunch of prompt templates across different domains and categorize all of the edits people made to those prompts and we came up with this taxonomy some examples of edits are maybe inclusion instructions or exclusion instructions a lot of people have very specific phrases that they might to include want to include or exclude which should be caught right in these assertion criterias so using that we can then seed the llm to help us come up with

assertion criteria customed to our prompt we can maybe use Trad TBT even to just copy in that taxonomy copy your prompt template and then say what are some assertion criteria that I should use based on my prompt based on these edits and come up with as much assertion criteria as you can and the LMS are pretty good at this they're slow but they're good they at least find things that are aligned with with you know mistakes that we might want to

catch via assertions so Spade first gets a bunch of natural language criteria and then generates a bunch of python function implementation I think the second part is less relevant like if you want to use a python function or a JavaScript function or you want to use the llm based validator whatever it is I think the the key idea to take away here is that you as a human are editing your prompt you have really good insights

into what the failure modes are and so how can you kind of codify that maybe using such a taxonomy into assertion Concepts that you should think of and we deployed this and we had a bunch of people try it out in a UI we deploy this with lanching so thank you to the lanching team for collabing with us here and we found that you know across the board across different fields inclusion and Extrusion assertions are most common and we found a number of

problems also with these assertions right who knows if llm generated code is correct we found redundant ones we found incorrect ones and then if you're interested in learning about how we solve those problems you can read our paper for more insights there cool now I want to talk about thinking about a UI around this experience

I mentioned you can you know use tat PT to maybe bootstrap some search and criteria but that's kind of underwhelming and

requires a lot of back and forth right maybe you go through Tatu P many times maybe you test it out in your jupyter notebook or the openingi playground you're jumping between different interfaces and trying to figure out how to make sense of it if you are a developer maybe at a larger company or trying to build evaluation tooling at your company right how do you think about interfaces for that so the main motivation for this actually came out of

spade just taking forever to run how can we you know use humans more efficiently in this process this how can we found that you know people wanted to improve and iterate on the Spade generated assertions and we they they also didn't feel like the assertions were aligned with their end goals or with their own preferences partly because they didn't fully know their preferences yet which I'll get to later but the goal of this interface here or thinking about an interface here is

you know how do you help people iterate really quickly and discover you know their own preferences on what are good and bad outputs and codify those into assertions as quickly as possible so key idea here is you know we've got to support we've got to minimize weit time when in this entire evaluation process so take a typical evaluation pipeline look something like this you've got a prompt that you're testing you've got a bunch of inputs out and outputs this latter

part of the pipeline where you're generating metrics trying to identify which metrics are good trust your own metrics and so forth that part takes a really long time so how can we include a human in the loop maybe humans can edit criteria refine criteria that llms come up with and maybe humans can also interactively grade llm outputs to figure out what are better prompts for the evaluators the llm based

evaluators so our interface which we also describe in the paper is built on top of chain Forge which is a prompt engineering tool and the idea here is to go through this kind of workflow the specific prescribed workflow you start out with saying I want to evaluate this prompt then you go to the B section here which is maybe I want to grade some responses first to look at outputs to determine what criteria that I should

include or maybe I want the llm to just view the taxonomy and infer criteria for my context whatever it is so you might go through that see the llm generated criteria or add your own criteria decide whether you want them to be implemented with llms or code then evalen takes you to a grading process which you give thumbs up and thumbs down on different examples and then evalen under the hood will determine you know what examples failed

what criteria so we can use them as few shot examples for those validator prompts and so forth Engineers your prompts there and at the end when you're tired of grading or you've graded all your examples then we show you like a report card of here are the functions we chose here are the alignment with your grades and then you can also scale those evales up those llm based validators up to all of your ungraded outputs with this table of

view so I like this a lot I really like that interface a lot like it's really cool because what happens is you start from your prompt and you know large language model is used to like look at your prompt and kind of guess like what kinds of assertions what kind of tests that you may want to write and then it helps like bootstrap that so it gives you a starting point it's like a it's like Rider's block it like gets rid

of writer block for riding EV vals it's really great I'm excited to show you the V2 that I'm gonna I a screenshot oh nice okay in here yeah so I also know I'm running out of time I can no keep going you want me to skip directly to that yeah sure if people have to go they can always watch the video and we've always run over so you okay sounds good keep going yeah great okay so speeding speeding away but

here's our interface and you know it's a research prototype it's super hacky breaks all the time so anyways we decided you know how do you even how do people even use an interface like this this is fairly new right figuring out how to help people assist people in coming up with evals for their tasks and an interactive interface I don't know if people have done this before but certainly there's probably a lot we can learn just by putting that interface in

front of people so that's exactly what we did we got 10 people we ended up not using one study but we got 10 people who are experts who have built lone based products and pipelines in production before and we asked them to use eval gen in an open-ended way we gave a sample task around named entity recognition from tweets a data set of tweets or they could bring their own task which I think like only one person

did their own task and we found you know generally people liked evalen as a starting point for their assertion so zero people thought the assertions that evalen came up with up front were good but they saw the value and you know unblocking themselves from moving forward and we realized that you know this evaluation and coming up with good evaluators is definitely an iterative process and people had a lot of mixed opinions on assertion alignment which we

dig into in the paper in more depth but I'll talk about you know two big things that I didn't really expect going into the study that I learned the first one is that we noticed as people were grading their own criteria for what is good and bad output is drifting it's a function of the output it's a function of the llm it's a function of viewing more outputs it's function of the rules that they include in the outputs

whatever it is grading outputs spurred changes or refinements to eval criteria so not only were they adding new criteria but also the participants were reinterpreting the criteria to better fit lm's Behavior so one a great example of This was there's this instruction in the prompt that says extract all entities from this tweet and don't include hashtags as entities so that was a criteria no hashtags as entities and we found that there were

multiple outputs that included hashtags as entity entities the first time people saw something like this they graded it badly the second time they saw the llm you know extract the hashtag as an entity they might say something like I said no hashtags entities but I think the LM did the right thing here Colin Kaepernick is like a very famous American football player and they thought that did something right and then when they saw this

failure mode

again for example Nik being extracted as an entity and they notice I'm failing everything I actually think the criteria should be no hashtag sign in the output and some people thought like the llm was smart enough to keep the hashtag if it thought the hashtag was like for example just do it the hashtag is part of the entity itself so it might include the hashtag in the output or if the entity is famous enough without the hashtag

then you know maybe it wouldn't include the hashtag I don't really know everyone had different opinions here which is the point the point here is that everyone has different different opinions and whatnot people want to go back and change their grades and people also have different opinions than what they had five grades ago so how do we build these interfaces and support you know this Dynamic evolving nature of what makes a good output sensing what is the

llm doing how can I make sense of it this is a natural part of grading human grading and the implications here are that grading has to be continual you've always got to be looking at your production data you've always got to be learning from that no evaluation interface we learned no evaluation assistant can just be a One-Stop thing where you grade your examples come up with evals and then push it to your Ci or push it to your production

workflow no you've got to always be looking at outputs and one of the things that we've been doing that's quite exciting at the startup that I'm doing ml engineering for is we have a slack Channel where we just log a bunch of outputs every single day for different llm based workflows and we literally look at them and we try to go back and reinform our assertions which definitely helps things evolve the second thing that we learned

from the eval study was that code based EV vals are very very different from these llm based evals I don't know why I thought going into the study they were similar but they're not people want to grade outputs to align the llm based evals but not necessarily the code based evils when they want to evaluate something like markdown format for example using a validator they just want to see the code that's generated to implement that criteria they don't want

to look at examples of good markdown not good markdown and hope that the llm finds a b example there and so when ask like okay when do you want to use LM based evaluators people want to use them when the criteria is fuzzy so they themselves like find it hard to evaluate or they don't have like a good idea in their head of you know what is good and what is bad they can't like succinctly describe it in one sentence but maybe by

giving enough examples the LM might learn something or learn that decision function for them and then also people want to use llm based evals when the data is dirty or there's typos in it so for example if Kaepernick who's a football player there's a typo in the tweet the output might you know correct that typo and a simple codebase function that asserts that the entity name it's in the input will fail because the typo was corrected but if you maybe use an

llm based validator then the llm based validator will understand that the typo is fixed cool so the last thing I want to briefly show you is that from these learnings we're doing an evalen V2 which hopefully I can and the idea here is how do we make you know coming up with EV valves a much more iterative process that doesn't just have this one step generate criteria

grade your done workflow okay so the idea here is keep your criteria as a dynamic list at the same pain as you're grading and when you grade you should be able to provide natural language feedback which might you know add new criteria might refine existing criteria definition and so forth

yeah I can probably skip this this one minute of video oh another thing that we found that was interesting which I didn't include in the slides but it's in the paper is that people want to give criteria feedback so maybe it's grammatically correct but not following some certain tone instruction so people want to give a thumbs up on one thumbs down on another so are you going to share where people can play with this is this public enough to where almost it's

very close okay we're all academics and like working so we move very very slowly this is a pet project but you can go to chain at Lea and you can play around with the table view that I had here this table view on the right if you write your own LL based evaluators like write your own prompts for criteria

yourself then you can run those and see this table view when you're prompt yeah I recommend playing with this I find it like whenever I hit a wall of like not being able to explain like writing evals to people I show them this and it works wonders thank you awesome yeah I'm excited for the V2 to come out but the V2 yeah I just need to implement some more algorithms in the market and I will do that when I have

time cool so this is my last slide My overall takeaways from all of this you know is when running LMS at scale there's going to be mistakes and we can use LMS to along with context for what humans care about for their prompts and what makes for good output for example prompt Deltas we can use that to assist them in coming up with good evils so there's no my slide animation okay cool yeah prompt Deltas

can aform assertion criteria and when you build an eval assistant it's got to be iterative is go to work and consistently solicit grades from the human as d and the LM prompts and parts of the pipeline as well and yeah if you have any questions please feel free to email me check out the pre-prints they're on my website and they're on archive thanks so much HMO for having me and Dan we've this is

excellent yeah the Discord the Discord is going wild they really they really love this stuff about you know these interfaces and this like explanation oh that's great yeah this is really awesome to see should we go through I I don't know St and I Eugene might be back I don't know if you guys have time to go through

questions I think we'll either way go through some of the questions that we have cute up here yeah let's do it I have like five minutes and then I got to eat lunch before the meeting I wonder I'm going to look through these

and see if there any that actually sh you can see some of these are there any that immediately come up come up to you as interesting ones you want to cover you can sort it by most most up

boats it's like sorted by time but you can change that got it like some of these already answered though yeah a good number of them are also for Eugene's ha would we get access to this notebook yes Eugene will probably share it yes all the notebooks are available I've posted a link on the Discord I don't know if H you can help

to pin it so it's all available there full of appendices as well I will tag you HL I create a try and tag you H so maybe you can help with pin okay yeah yeah please tag me okay I found one question that is up voted a lot and relevant for me using prompt history to generate assertions is very interesting believe this can be used for unit tests llm as judge assertions it's a goal here to improve

assertion coverage and reduce the time it takes to write these assertions by hand okay this is a great question so the first thing about having the prompt history is that it focuses the lm's attention when you're asking the llm to generate criteria for you if you just past on your prompt into chat and ask it you know what what criteria should I design unit tests around chbt will just design a

unit test for every sentence in your prompt which maybe that's something that you want chbt is very verbose it just comes it just recalls everything but I find that you know however ring like 15 criteria especially for long prompts is like a little a little bit much I want to start out with like two or three that I feel like are good ideas and really work on you know fixing those criteria making sure we have good

implementations of those before adding new criteria in this case then providing the pr history the Deltas right like it's a great example of what you care about what the llm is bad at this can focus chat gbt or the attention and generating assertion so maybe it'll only come up with four or five eval criteria which is a much better starting point I think than 15 hopefully that answers part of the question the reducing the time it

takes to write these assertions by hand I don't think the I don't think it's extremely difficult to you know come up with at least one or two good assertions what I think is really hard is coming up with assertions that align with what you think is good or bad this is hard because you don't even know what you think is good or bad like you have to look at a bunch of outputs to be able to

Define what's good and bad and that process also evolves as you deploy things right you users your users might complain about things that you didn't expect and then suddenly you have to also now incorporate that into your definitions of good or bad so having kind of an evaluation assistant to help you draw conclusions from all of this constantly changing data help you define what you think is good that's where I think the biggest value lies not

just like generating code from an lolm yeah I agree with that there was a question that I don't know if it was answered or not from Wade how are you organizing unit tests where are you running them oh no Eugene did answer that sorry I think that yeah I

think that was a question for Brian got yeah and he mentioned something like notebooks as unit tests orchestr oh yeah he talked about the superpower of hex for running this it's my favorite subject which we won't get into from a former life okay so other questions oh there's there's some I can do pretty fast like how well do

these assertion criteria attracted from p PS in the prompt versions generalize across models pretty well the data set of prompt edits that we looked at had prompts that were intended for mistol llama 2 chachu BT three 3.5 as well as four and Claude 2 so I don't know I think people make edit their prompt in similar ways no matter what LM they're using

there's a question from Lucas in the honeycomb example it was clear that you could write good unit test for the data because you know is the query valid etc etc but I imagine it's a lot more difficult for the general llm input output pairs curious to learn more about that so okay generally speaking in an applied use case I find that more often than not it's narrow nrow enough to where it's not just

general language like replyed to anything do anything there's some specific task that you want to get done you're trying to Aid the user to do something specific and a lot of times like there's a lot of components like there's function calls there's rag there's something like that and there's a lot of failure modes that can happen that you can test for however like Dan as Dan mentioned there's some use cases where it doesn't really like if it's just like an internal tool

where you're just trying to do something like reword text or summarize text or you know there's definitely use cases like that where maybe unit tests are not going to have as much teeth but it's always good to think about you know these unit tests then you see any of these questions you want you think are interesting so a lot of the top ones are about

function calling I and agents I think Eugene sort of answered on function calling an agents I might Mark those as as answered unless yeah anything else to say there we got somewhat heavily uploaded and there's a quick answer open AI has a temperature program in their API is there something similar in open source llms that we have to account for yeah

and unit test yes so these models also have a temperature set it to zero it's actually some class models where get an assertion error you get some error if you get set it literally to zero and you set it to one minus 6 or something but yeah all the open source models you can set temperature to zero as well I thought this question is good it says before before before starting a task how important is it to have

evaluation method how to fix this as you learn more about the task in the process of doing this okay so you don't need to set up evaluations in the very very beginning like make some minimal product or something like you don't want to just be like super academic and say oh I have evals before like before even beginning you kind of you might not even know what you're trying to build like as far as like when I build stuff like I don't

necessarily know all I don't really have a clear picture I kind of have to like shape it a little bit and so you can certainly start with like without evals but you need to quickly think about evals at some point when you want to improve the system and so don't let evals necess get in the way of making something just know that like it is a like a crucial thing when you're trying to at some point make it better

and that's how you make it better and to carry that even a step further I think like one of the things that not thought about very crisply but really came out of TR talk is that when you see you actually don't upfront know all the evals you want and one of the nice things about working with llms is like to call chat G like to open up a browser window and try something and see what isn't working is super easy so I would

say you probably don't want to do anything formal upfront like run a few examples like literally type ad hoc in what you think the question is into CLA or chat gbt then be like here's the thing that's not working and that I realize is difficult and you'll do a better job once you've just experimented around and then build up the complexity over time rather than after you like experimented rather than trying to imagine the pitfalls up front

yeah I agree I just want to reiterate what H and then have been saying I mean we don't want to get into eval paralysis like we don't need have all our evals laid up front before we start doing something just build something small 30 samples maybe would be good enough and as you start doing it you start to see more and more edge cases and that's how you add evals Right add evals like test cases so it's an iterative process so

we don't want to have evals also slow you down slow down your building process it's it's good as a test hunus it's as an insurance it helps you stay safe but it shouldn't slow you down and it's also interesting to think about so there different use cases or different problems some the eval is actually like just classification and for those you know Eugene showed a bunch so showed examples the eval it's pretty obvious what the eval should be and then there are others

where it's not classification you're generating free form taxt it's like very fuzzy and those you'll accumulate vals iteratively as you see like what doesn't feel right yeah another one that's highly upvoted but I think has a simple answer is LM as a is LM as a judge a fine-tune model or only works by improving prompting I think LM as a

judge there may be exceptions but it's almost always just like a very very good very very smart publicly available model because if you're fine-tuning as typically a case where like you have a bunch of examples of the correct behavior and you might use that to build like the model that that outputs the the data but the though number of ways these models can fail is sort of like open-ended and fuzzy so I would say I've only used LM as a judge with models

that are not fine-tuned and I think that's probably generally true yeah I mean I have only fine tuned a model once for this in the specific use case I can't talk about but Mo I think I would avoid it because then becomes like turtles all the way down or like you know yeah like basically you want to try to use off the shelf model and align it with with the human because because like the complexity is like way

too high if you start like fine-tuning this other model it's like judge model it becomes insane and I don't recommend it how do we go from 0 to1 instarting the data flywheel for collecting user data and curating the data set I actually think this has a similar answer of starting the great thing about llms is that there are some very good ones off the shelf there's no fine-tuning you don't need data I would start with a prompt

and you can probably if you want to build a data flywheel like implicit in that is that you're going to build some product that you're going to increasing usage on and then collect data but I would actually start with just a prompt and for most problems you can write a prompt and use a generally available model that's reasonably good yeah synthetic data generation that's the whole Magic of LMS you can unblock yourself a lot of times not every time

but fair number of times your code here's one with three up votes your code uses do sample equals false but real life prod will use do sample equals true what's the thinking here it varies a ton by use case but I would say that for the use cases I deal with do sample is pretty much always equal to false do sample is basically like is temperature nonzero we like our for the products

I work on we just say like we want something deterministic we want the best answer oh we don't need variety if you were building like character AI you would want it to be more creative and and varied what about you guys do sample in prod is that usually true or false you mean for f shot examples no sorry the the do do sample parameter in

when you make your generation call no I haven't had a reason for that yet or do you what's your temperature is it zero or nonzero in prod I think mine is zero most of time that be that'd be the same as do sample equals false and that's always been the case for me Alterna take to this I usually start with 0.8 and then I lower it as necessary to

achieve forever performance I think there's another heuristic I've heard people say which is you start you want to get as close to zero as you want as you can for classification or extraction tasks and you want to get as close to one for creativity and generation task so that could be that could explain why I'm closer to 0.8 but essentially if you get too low it's almost like a dumb inter and it depends on on what you want to do

so do try that out the crazy thing is for open AI the temperature max is two I don't know why don't ask me why so if you if you if you're thinking about temperature that's something think off you if they let it be high enough it just be a random token generator exactly

there's a comment from I probably can't even pronounce there or question from I probably can't pronounce their username man manic when doing ab test on an LM how would you prepare the data what I mean is do you ask just ask your LM to vote AB or do you do some prep ahead of time there's a chance that that was mis misinterpreting something that I talked about earlier where I said we're using AB test we actually have two different

models that are producing output so in this case that was for alt text like two different models that you could use to take an image and get a description from it and then we had people who rated each of the some people would rate one model and some people would rate another model and then whichever got higher scores we just like okay that's the model that we're going to continue using the model that got worse scores we would just

discard but it was the people rather than the model that were assigning scores you could in theory have an llm pick between two candidate pieces of text I've never done it and I don't immediately know the use case for it and then it would be hard to answer this question of like how much data cleaning do you do before that if without I think it would always depend on the particular of the problem and why you're using an LM for this

so-called AB testing let's go back to the top of the uput can you talk about what metrics to use to evaluate retriever performance in rag I think there's probably a better answer than what I've done historically Eugene do you have any thoughts on

this one yeah a few things I won't go into the standard ones I think the first thing is let's say if you your contact size is like 10 you want to make sure at least you have some of it that's relevant you have some relevant stuff in there that's recall and then there Al that's also ranking but what is also quite important so recall is really just recall at 10 how many of those documents are relevant

and ranking is you want to make sure that the more relevant ones are closer to top personally for me what I find to be quite important for rag is this metric that I've never had to considered before and this metric comes about what this metric is is how often can you actually return zero results if the customer is asking a question that you have no documents for so if you're using purely sematic search sematic search is

just K&N you just go grab whatever is the nearest and the similarity could be 0.1 but you just pull it out that's C the problem this happens is that LMS just cannot distinguish relevant from irrelevant data the more recent ones can but it's just not very good at doing that and a lot of this is also because of this eval which is called Nether and Hast stack that forces the LM to try to pay attention to everything

and try to use it I believe that there's an alignment TX where you try to optimize for needle on a Hast you also reduce the lm's ability to reason over irrelevant tic documents in the context so that's why it's very important to me that hey we're going to have some test queries that have absolutely no data in our retrieval index and we want to make sure that it's always we get zero or close to zero so long story

short recall for one just to make sure you have at least relevant data and that's recall at 10 ranking which is ndcg you want to make sure that your relevant data is closer the top and then also I don't know what this metric is but ability to return zero results especially for queries that you have absolutely no data for so that you don't return trash and you can deterministically say that there's no answer instead of letting LM say that

there's no answer if your if your retrieval results is zero size of zero you can DET deterministically say hey I

don't know and you don't even have to make a make a stupid mistake in front of your customers yeah that's a good point and it reminds me when I I talked about this project I did in Workshop one I talked to this project I did for this chat bought for a European shipping company called dpd and they had

someone asked the model a user asked the model to like write a ha coup about how crappy the company is and then this person published it on Twitter and it got picked up by the news and it was like embarrassing for the company when we said like how are we going to fix this and make the system more secure this isn't a perfect solution but one of the things that we did was to say if you don't have any document that meets

some relevance threshold there's a good chance that actually this is a user who's trying to do something that we actually don't want to allow and so it's not even like that we can't do a good job of it this was just a way of detecting adversary like a not perfect way but a way of detecting adversaries and shutting down shutting down that whole interaction and the amazing thing with lexical retrieval lexical retrieval can have a score and even embedding retrieval you

actually have a score in terms of distance right you can say if anything that's way way too way too low on similarity just threshold it yep that's beautiful this one I think is for you the top voted one from Sam silver so Google had a curfuffle where they retrieve relevant documents but the documents were not factual right like the description it didn't hallucinate relative to the documents it

hallucinated relative to reality I'd be curious to hear about the relative importance of these problems and if you've ever worked on filtering documents the documents or the Corpus itself to ensure that the documents themselves are factual or unbiased or whatever else that's a great question and sometimes this happens sometimes your documents could be factual but could be bias it could be about racism fascism whatever and you know there's documents out there like this I don't know how to solve this

problem yet I think you could probably solve this problem with a content moderator over all your documents like say if you check if your documents are I mean the ones that very clear is like toxicity bias offensive Behavior not safe for work sexual content those you can easily do and because you're using rag you can very easily exclude them from your retrieval index that's your immediate ending C right so imagine if I was Google's case you know the pza and the glue okay we

know glue is causing it we pull the endem cord remove that piece of that piece of data from your retrieval index so that the Google search AI summarizer never sees it problem solved I think that's how I would solve it but as to how to actually check this kinds of data where it's clearly Miss leading or clearly untrue data I actually don't know yet U if if we have some data that we can learn I I think content safety is R

straight for offensive data is R st for we have a lot of data on that but for things like this that's really about a threshold that we've never had to Grapple with before I I think it's still an open problem are you running unit test during cicd it could just take very long with non mocked llm calls I think that I'm supposed to be running it in cicd but to be honest the for so I was quite interested so Brian

said that the purpose of most evaluations is to let you be able to sleep at night and for unit tests there's probably some truth to that for me the way that I think about it is actually quite different from Brian in that there's a thousand different modeling choices I can make what base model do I use what's my prompt what is the context that I feed in do I find tune if I find tune

what's the data that I find tune on I could make arbitrarily many models and I need to decide which of them are better and which of them are worse and so if it's just for making decisions and a lot of those are like we want just a quick decision and then we're gonna make a change to the prompt and then we're gonna run this again and see is it better or worse and we're gonna make another change to the

prompt and so frequently I just want it to be really easy and low latency and for that reason we typically run them the developer runs them locally you don't even need to push anywhere if we wanted it to work like conventional unit tests of like a safety thing you know haml gave the example of not exposing private data then I would probably put it in cicd to avoid the possibility that we forget to run it on something that

gets deployed my use cases aren't like that and we're just measuring quality and so we I've run always run it locally and then yeah can take a long time yeah you guys have a different answer for that no that was a good answer all right

o I like this one from LZ good ways to check for this is by three right here but are good ways to check for contamination of Base models with existing eval data using paraphrases for example so you're going to test your model you want to assume that how it does in your test is a good proxy for how it will do on new data how do you know that your base model wasn't contaminated with the same thing you're

going to use for evaluation I'm GNA answer first yeah I don't know go yeah I mean it's like it's kind of like very similar to machine learning in general like okay it is useful to kind of also look at your production data and see whether that is like skewing really bad relative to like your validation data and whatever ever that's a smell that

you have some kind of leakage another smell of leakage is it's too good leage is hard to be honest I don't neily have bulletproof defense for it this one also seems super context specific so for instance let's use Hamill's honeycomb data as

an example there could be some honeycomb queries that are in the gp4 training data there probably are but there's no reason to think that the ones that was collected for him to find tune on are more likely to have been pulled from the gp4 training data than ones that they will have in production and so there like you can just sort of reason about it or if you use my example we had that I talked about today was this

debiasing essays those were essays that just got written or not essays we call them Journal articles they just got written they were submitted to an academic publisher and now we're going to edit them the fact that they were just submitted for the first time literally like days before we came to edit them would make us think they

probably weren't in the training data so I think this probably happens sometimes and you just have to it's probably just

I don't think there's a general rule for how you avoid it should we go for you want to go haml I think we can end it okay yeah that's close it here anyone who we've got 160 people left actually before you guys all dropped drop off for our Discord server we we are

rotating links I should figure out if there's a way to make it not have links expire every seven days but email me if you have an outdated link I want to understand how many people this affects we have like a selected sample here sorry and then please let the form redeem your credits and then I think we've got some good sessions lined up for tomorrow and basically the rest of this week

thanks haveone