Inspect, an OSS Framework for LLM Evals

46 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=KNAZU9BZ-UM
https://www.youtube.com/watch?v=kNaZU9bz-UM

SUMMARY

Inspect is a Python package designed for evaluating AI models, developed through collaboration with the UK AI Safety Institute. It provides a flexible framework for creating and running evaluations on various datasets, including tools for debugging, scoring, and visualizing results.

- *Inspect allows users to create evaluations using datasets, solvers, and scoring mechanisms, facilitating a modular approach to AI model assessments.*
- *The framework supports local development and can scale to cloud environments for larger evaluations.*
- *Users can integrate various solvers and scoring methods, including custom implementations, to suit specific evaluation needs.*
- *The package emphasizes interactive development, enabling users to work in notebooks and run evaluations in CI/CD pipelines.*
- *Inspect supports a variety of models, including those from Hugging Face and custom APIs, allowing for extensive flexibility in model evaluation.*
- *Logging and diagnostics are integral to the framework, enabling detailed analysis of evaluation processes and results.*
- *The framework is designed to be extensible, encouraging community contributions and the development of additional tools and metrics.*
- *Future developments may include enhanced integrations with tools like Weights & Biases and expanded support for various evaluation metrics.*

so I'm going to try to give a kind of Whirlwind tour of inspect and I actually worked with Hamill a little bit over the weekend to build some inspect evals for the honeycomb data set that you all are working with so hopefully it'll be it'll connect well to the work that you've been doing and it'll make it the concepts gel a little bit bit easier so what is inspect not surprisingly it is a python package pip install and spect AI someone

asked a question about and Hil reference that I'm that I'm affiliated with posit and which is formerly our studio this project is actually not a posit project this is a project that is results from a collaboration that I'm doing with the UK AI safety Institute so the UK AI safy safety Institute has hundreds of evaluations of every variety you know simple QA Things Fancy Capt you know cyber security captured the flag evals hundreds of evaluations

and when we kind of started off on the on the journey to writing all these evaluations there weren't weren't terrific tools available a lot of the tools were either embedded in Benchmark Frameworks or maybe not very complete or weren't that well tooled it wasn't necessarily clear exactly how much more development they would get and further I think they were not necessarily designed to scale up to very complex evals and so we we set out

and actually the most popular eval

framework is just roll your own eval framework which was out there a bunch too so we set out to build something that we could use and that ultimately we could share so so other people could use as well so with that just again trying to ground this in honeycomb this is actually an eval for the honeycomb data set it's got I think there's 2300 user inputs we've we've also got the you can see columns that's the

schemas that were that were fetched originally by rag so they're in the data set and so to create an evaluation we basically take that data set and then we put it through a pipeline which you'll recognizes really the same code that Hamill had in in the notebooks for for honeycomb and then we apply score to it which is again based on that Hamil's original code and then we can run this eval and get lots of tooling you can see

this I'll get it more into this but have a log viewer that lets you explore and debug and visualize everything that happened during the eval lots of other tools this is a vs code extension that lets you tweak things run different models Etc so I can I think I would emphasize that inspect is not super opinionated and really is like python code first and by sort of conforming to some simple conventions and fitting yourself in you get to take advantage of

this big pipeline of tools and then hopefully like a big ecosystem of related packages that will extend inspect so I'm going to go is is it a very local in nature like it's local it's all local yeah okay yep yeah so one of the things I skipped over on the first slide is we have this concept of development to production so we definitely want to have like a very interactive local you work in a notebook like be very iterative you know work on

your eval we have lots of stuff for taking it and like running in the cloud you know run it over 10 models run it you know so it's it's like local for development but then we have lots of tools to if you want to scale it up and run it as part of CI and things like that so but it's all it's all local very okay so this Core Concepts and you saw a little bit of these on the

on the first slide you have a data set which is not at all surprising what that consists of it's inputs and usually there's Targets in the case of the honeycom data set there's not a Target there's just there's there's going to be a validation function and and a critique model Target would usually have like if for multiple choice what's the correct answer for Q&A some kind of description of what the right answer is for a fancy model graded eval it might

be like a grading rubric for a model so that's the data set solvers is like the pipeline that actually does the eval and this could be doing anything from prompt engineering to calling the model to a multi- turn dialogue with the model it could be doing various types of elicitation like critique solvers is kind of like the heart of the entire eval and where you kind of customize how it works and then the scorer basically evaluates the final output again these

can be very simple like doing text comparisons they can be model grade model graded or they can use all kinds of Customs custom schemes whatever kind of whatever you can dream up so that's the Core Concepts and I'm going to drill more into this validate example so this is the eval I'll break down the different parts of it we talked about the data set and this is just reading from the CSV there are standard fields that go in data sets input is one

of them and then there are like custom fields that different evals will need in this case columns is important because we're going to use columns for the prompt template and so we we want to save the columns when we read the the data set in and then this we have this plan and you'll see this is the same system message used in the notebooks presented in the course prompt with schema is a solver that's going to

build the prompt that uses the columns and the input and then generate calls the model so pretty straightforward and then the scorer actually does the uses the check function the check query function that you've also seen in the course so that's the eval and I'll drill into some of the specific components they're they're quite simple and I hopefully they'll be very intuitive and straightforward what they're actually doing here prompt to schema is literally just taking a prompt template and then

substituting The Prompt and the columns so you've seen this before but it's really it's a solver that really just makes a prompt that's all it does and then the scorer is basically going to it's going to take the output from the model this Json completion function is just sort of a helper that some models actually like to put like Json like code blocks in so it like strips that away so the idea behind that is

just like clean it up so we get pure Json read it and then we call the is valid function which is literally the exact same is valid function that's that's used in the course and we figure out whether the U whether the query was valid okay and then so once we've done that we've built our solver and we've built our scorer and we've run it now we run our eval and we can see kind of what

happened and in so many evals it the score really doesn't tell you anything close to enough especially when you first start developing them because you know your your checking function could be wrong your the way you extract answers from the model could be wrong there's so many things that you need to investigate and you kind of in some ways need to do it on a per sample basis so this lets you very easily look and see

what happened overall and then what happened on a sample by sample basis so here you can see drilling into this sample which got incorrect we can see what was the actual message history was the dialogue between the the model and the eval and so here it's quite long as you know from working on it here's the columns that was injected the schema and then here you can see to the very end and then the assistant answer so looking at all the messages

can be quite quite valuable especially when you get into things like like tool use and then also okay so that's that's the basics of that then I built another eval which has actually it's the exact same code and in fact in the in the git repo that I'll share at the end you'll see that I do reuse the code I don't I don't just copy and paste it but really there's only one line that's different here it's the same

data set it's the same plan but we're going to use a critique model for scoring instead of the validate function so here this is the critique scorer so this has a little bit more going on but it's again pretty straightforward notice we parameterize what model is used to do the critique so you can use you know it's pretty obvious but you don't necessarily need to use the same model in fact you often don't want to use the same model to do scoring as the

model you're evaluating you can use a more powerful model in some cases maybe use a fine-tuned model here we're defaulting to gp4 Turbo but the user of the score could could use any other model they want we build the critic prompt kind of analogous to how we built built the other prompt and again this critique do text is literally the same it's the exact prompt that U that is in the notebooks for the course and then we run the critique so we

get the model instance we call generate we parse the output we check and see if it was good and then we return the score so that's our critique score and then at the end of that we get another eval view this time for critique the the accuracy here is a little bit lower and these these numbers actually don't really mean anything by themselves but just noting that it was less frequent that the that the critique was satisfied with the

output than that the the dev validator which is intuitive and so here we might want to know what actually happened in this critique so we can drill in here this is one of the incorrect answers and we can see what the answer was and then we can actually see what the critique models explanation was so here you might you might look at it and say wow that is actually looks right to me or a human expert may

have said it was right and then you want to look at the explanation and perhaps this indicates that you need to improve your your prompt template for the critique model or perhaps it means you need to fine-tune a model just to do critique depending on how many you know what what resources you're willing to apply to get a good grader so this is again just Drilling in and seeing what happened during scoring okay so that is there any way to

interact with this this view like do some annotation in this view itself or something so right now we don't have we don't have like commenting in the view and stuff I think we we are developing like some like shared places to look at like internally places to look at views together and there'll be some annotation in there I don't know if we'll come out with like a thing to do that but that is useful especially yeah so maybe there's a way we can do

that in a way that that we could open source that as well and then and then people can take advantage of that this is just runs locally so you kind of would need to host it in some kind of we've talked about actually building a it wouldn't really solve it a weights and biases plugin but that's not going to let you do this like drilling into each one may maybe to some extent it would if they have like good good

templates for for for chat message for chat conversation histories but yeah I like the way that the this is being rendered and it's flexible seems flexible enough that you took my example Without Really knowing it until the weekend and seem to just plug it in here and it's actually just plug it in and mostly just use all your code I mean there's a very what I showed you is like all the code that I wrote and all the

rest is just calling your using your templates and using your code so that's kind of the idea that you you know you there it's a minimal lift to get your existing stuff working inside the the pipeline so yeah we did that okay so now I want to just just talk a little more abstractly we've looked at at these examples I've shown you some simple solvers really simple just a prompt template they get they get a lot more

fancy which I'll show you in a minute and some some scorers so like conceptually what is a solver it basically and this is a simple view but the idea of a solver is it has a task state which is like what's the current state of the message history and what is the current model output and when we start there is no model output and it do it just transforms the task state in some useful fashion so that could be

calls the model gener appends the assistant message and updates the output it could be prompt engineering it could be critique it it kind of can be anything and yeah that's sort of what I'm saying here like the Sol function does something useful with the state and that sort of allows you to create a pipeline of solvers now and reuse solvers it's not always the case you want to do that some evals really just want to have like one solver that

just is sort of boss of everything and doesn't confine itself to being in a pipeline but but the pipeline does end up end up being useful in a lot of a lot of cases so some examples of really simple solvers which you've sort of seen hints of we built a custom prompt template for for the valid for the honeycom evals but like a simple here's a prompt template just transform the prompt by passing it through a template

perhaps with some extra parameters from metadata gener this is actually the actual source code for the generate solver it literally just calls the m to generate and that's it so those are like really simple 101 solvers I think we have like a Chain of Thought solver which what does like a basic Chain of Thought template but even then often you you want to customize the that for your for the domain the generic Chain of Thought isn't always

exactly what you want and I guess this emphasizes when you're writing really good evals you are writing you can reuse solvers but you're oftentimes want to write your own solvers and write your own scorers to to make them really really good here's a more fancy one I won't actually walk through all the code on this but it's a multiple choice solver that handles like shuffling the choices and then it actually calls the model to generate and then it

unshuffled the choices so this is an example of a solver that calls generate in internally So the plan for a multiple choice will typically just be like plan equals multiple choice it might be like plan equals Chain of Thought multiple choice where multiple choice just calls calls generate internally another one self-critique pretty straightforward again we've we've already called generate before self-critique comes on the scene so we may have had a pro a system message a prompt template now we call generate and

now self critique so the first thing we do is we actually take the existing completion and we we basically run a critique on it and then we take that critique and we append it to the message history and then we call generate again and so this is an example again of a solver calling generated internally this time just to reup the answer with the critique being available I love how you made these slides I think with quto there's a lot

of comments in the Discord no okay yeah yeah yes with quto definitely with qu oh yeah yeah yeah yeah yeah yeah yeah totally yeah a lot of comments in the in the Discord about hey this is really cool is it Cordo so yeah yes definitely it definitely is Cordo yes yeah okay and I'll make the slides available at the end and the slide source code and everything so there's a get repo that I'll that I'll share a link to at the

end so people can figure out how we did all this okay so that's the self-critique solver okay and then there's generate we already talked through that okay so one of the things to think about is like is composition and so one of the ideas behind inspect is that you actually will write people will write a bunch of python packages that have scorers and have solvers and that you'll be able to mix and match those so the idea is like

there's like ex you know lots of external components to let you do different things that you can just plug in so as an example from from AI safety Institute we have an internal package called Shephard that's used for doing jailbreaking and so there's many there's as any of you follow the literature there's like dozens and dozens of jailbreaks and some of them work and some of them don't and sometimes you have to try multiple jailbreaks and some

of them work with some models and not with the other models etc etc but these are basically jailbreak solvers that essentially do prompt engineering to get the model in a in a state where it may it may provide answers it wouldn't it would otherwise refuse and so as an example here's a eval that's basically trying to give them it's basically just trying to get see if the model can give good good computer security advice so it's like how do I prevent you know

you know my website from being hacked and then so sometimes the model will because you're asking about computer security it'll it'll like flag don't don't don't talk about computer security and so we're saying well we want to we want to see what the model actually knows and so here you can say we bring in a jailbreak solver from Shephard and then we just use it in our Pipeline and so we have our normal system message we put the Jailbreak in

and then we'll probably be able to elicit more or have less refusals than we than we otherwise would and so you can imagine lots of different solvers that you could that you could plug in prompt various types of prompt engineering solvers you can imagine a whole like python package just full of of prompt engineering techniques and a whole python package full of critique debate all kinds of things like that so and similarly scorers so python package is full of different

variations on model graded scoring and things like that so that's an example of composition which we think will be a big part of of how people end up using inspect okay okay so i' I've simplified a little bit where we've just been looking at really straightforward kind of QA Style tasks or in the case of of obviously honeycomb we're looking at a fine-tuning task but these are straightforward just like prompt generate nothing fancy going on I

didn't show you all of Tas State before but Tas State also includes tools and so the idea buying tools I'm sure you've all seen or used you know these are python functions that you can make available to the model and you

tell the model you write the python function you write the doc string you tell the model about them and then it will say hey I'd like use this tool and so part of the task St is a list of available tools as

well as potentially like a nude to the model like definitely use this tool or definitely don't use the tool Etc and so this is a simple example of this is a QA T biology QA task and we're saying hey if if it tends that you don't know the answer I think this data set actually has a bunch of very obscure questions then hey you can use web search and so then we have a web search tool that goes and you know gets a bunch

of Google hits and summarizes them and things like that and so use tools as a function that just makes tools available to generate and so there's you know once you get into tools now you're into agents sometimes it's just simple tool use like hey let the model use Wikipedia or let the model use web search and sometimes it's given it all kinds of tools and they really become agents at that point and so you can have agents sort of with like

very bpos custom logic or you can bring in like an agent Library so I'll show you an example in a little bit of taking like an existing Lang chain agent and just like basically making it into a solver so the idea is you can just take a take any agent in these other Frameworks and once you have the bridging it'll just work inside inspect so I'll show that show that in a minute but may first show the sort of

bespoke agent concept which is this is a a cyber security eval it's a capture of the flag task and this is more like the hand red agent Loop where we're basically giving a knit challenge is here is is creating a Docker container use tools is basically just like here's some tools and we tell the model like you can do all these things and then we give it a task and then this is basically just a loop where the model

gets to keep using tools until it either terminates because it didn't couldn't figure it out or it ends up finding the flag so this is like very custom this is like roll your own agent and definitely that's a thing that's something that people do but at the same time you know there's lots of agent Frameworks out there and so we want to be able to have like high order functions that let you take an existing agent framework and

turn into a solver so this's an example like if you look at the code in the middle here this is all code that is just Lang chain code there is actually I haven't shown the Imports but there's no inspect code at all in here this is just like code that you would be writing in Lang chain this is basically going to get the Wikipedia tool and then using the tool and then we have as I said

there's this higher order function this is actually provided as an example right now it's in the repo that I gave you but we can just take that agent and if it conforms to like the Lang chain agent interface we can just turn it into a solver and then what you if you actually look at the eval that results you can see this is the Wikipedia search so it's a data set of can the model use use Wikipedia to answer you know they

may be difficult questions may be obscure questions there may be questions we think the model definitely can answer but the idea is you know the plan is literally just this Lang chain agent that uses Wikipedia and as you

can see down here is the solver that I showed on the previous slide this is the entire task definition use a model to assess it and use the agent and then you can see here kind of what happened and if we

look kind of inside at you know what happened during the eval you can see it it'll show you the tool use so it'll it's like okay what was the back and forth what what tools did the the model choose to use and why it gives an explanation what result did it get and this Game of Thrones one is really one of my favorites because it ends up it's it's it's trying to find the the in order of the 10 episode titles and often

times like in Wikipedia it's not like literally spelled out or actually where it is spelled out it might be wrong and so of times it'll do two or two or three queries to try to sort it out so anyway it gives you that's sort of Diagnostics of what happened with tool use and then similarly for scoring this is the model graded fact scorer this was the answer and it was incorrect so this was this was the grading guidance

this was the answer it was graded Incorrect and I think I'm going to hopefully show you yeah this is the scorer explanation so again you know sometimes the model really didn't get it but sometimes the score is actually wrong and so it's important to be able to look you know drill down and see what's what's actually happening okay so that is okay so let's just talk a little bit about scoring I'm just going to check my time

and make sure okay we're good a little bit about scoring there's lots of different ways of scoring obviously traditional like pattern matching and template and answer based scoring or obvious and we've got lots of built-in solvers for doing like regx matching matching at the beginning and the end matching a a template like where you tell the model to say answer and then the answer lots of that sort of thing there's also model graded scores built in but usually you need to customize the

templates for those to get them to work properly for your domain and of course as I mentioned before like they're pluggable you can get them some other packages and I'm I'm expecting lots of stuff is going to happen with model graded scoring over time and we'll see we'll see the the benefits of the community working on that over the over the next months and years and then you can also just say no score have a human score it so

that's that's also possible and one of the things I think and this is something that I know is emphasized quite a bit in the course is basically rigorously evaluating monograde scores against human baselines basically I've observed that definitely a lot of people will get their model graded scorer going and they'll be like cool now I have a scorer and they haven't actually grounded it and whether it's how good it is relative to human scores so if we

can build tools that help people do that well that sort of structure that work I think that'll be that'll be valuable so that's something we're definitely going to work on okay so what am I oh this is okay this is scorer example which I think is pretty interesting this is the traditional this is like say the math benchmark that I think open aai reports as part of their standard benchmarks and what's interesting about it is that the model does math and

then there's a Target but often times the answer is correct even though it's not literally the Target and so we

have this expression equivalence solver that basically lets a model assess are those expressions actually logically equivalent so it can even do a little bit of algebra or a little bit of factoring these are trivial you can see this this is the same as this it's scored correct this is what's going on in the the that equivalence thing is

it a redx or is it more in there there's more going on I'm going to show the full source code to it in the next slide there's a red to extract the answer and then we're going to go and have the model so so we we prompt the model to basically say at the end put answer colon and then the equation and then we basically that's how we pull the answer out and then we send that to the to

this expression equivalent solver these are trivial because they're just like punctuation differences but I've seen it where it actually can sort out that the expressions are actually equivalent so let's take a look at closely at that at that solver hopefully I have a little step through on this now I skipped through the okay so extract the answer and this is a reg X off of this line answer pattern which is a common way of prompting to get to to get

the model to delineate their answer in a way that it's easy to pick out and then here we actually have a whole another template which I'm not going to show which basically it's a f shot thing that basically has like it has like 20 different few shots of like these are equivalent these are not equivalent and and then the model is able to take those and then actually do do a pretty good job grading pretty surprisingly good

job grading and so this is you know you kind of this is a custom eval it's math equations you have to build a custom scorer you have to use a model to help you do the scoring but it kind of gives you a taste of some of the things that people will do with scoring okay I want to talk a little bit about what might seem kind of a mundane concern but logging ends up being like

massively important for doing good eval obviously we build a log viewer on top of the log but the log also has an API so that you can interrogate it and you know you can get multiple logs and then plot the differences in things so the idea is the log is a rich rich python object it's also Json there's a Json schema for it but it's a rich python object that lets you explore everything that happened during the eval and

compare logs and things like that so there's that and then I think you've seen most of the examples of the log viewer but showing the samples showing the messages yeah you've seen this showing scoring okay so that's log so a lot of people the other thing people do I think I'll show this later they'll run like 20 eval tasks like doing a grid search and then they have all their logs and they and they plot the the results things like that so

definitely like you end up Computing on the logs quite a bit it's very cool yeah so models we support a lot of models we do you know the big big Frontier labs and do you need to support like specific models like is what's the difference between using like any hugging face model can you just use any hugging face model really yeah it can it can absolutely yeah no what it is is this prefix here that is the model

provider and then this is completely arbitrary so this is like any hugging face model this is like any model that

you have locally with oama this is any model that's on together AI so this is provider and this is model name we don't know any I guess to answer your question we don't know anything about these model names we don't resolve them we don't compute on them we don't know what they are they just get passed

through so when janim and I you know 2.0 comes out you just start using it so yeah and can you have your own endpoint like your own rest API endpoint yeah you can yeah so one of the things that H is interesting like oyama and VM both actually and together I think together might use VM they all use open ai's API so sometimes people will just you know use open AI with a custom based URL but

you can also you can create a custom model provider as well if it's like completely custom rest API that we don't know about it's very easy to to make up model provider and publish it in a package or or what have you so yeah so you should be able to get to the models you want to get to without without without trouble cool okay okay so let's see I just want to make sure we have time for questions there's

a there's I'm gonna go a little fast on the rest here but just to say we care a lot about interactive development we care a lot about being able to work in a notebook doing exploratory work on the eval but then we want the eval to end up in a form you can like run them in CI you can run lots of them you can systematically compare results and things like that so we have good like we

have tooling that works well in notebooks you know I showed you before you saw like a terminal it was like inspect eval do the thing you can do all this in Python in a notebook and it does all the things in the notebook you can have tasks in notebooks and so we we definitely try to invest a lot in like interactive workflow and then make it so it can scale to the more I would say production

workflows so again I'm not going to dwell too much this is like a grid search so it's like okay I'm doing a grid search over different grer models graders and system prompts and that product is just like a thing that's making the grid search I'm dynamically synthesizing a bunch of tasks and I'm going to run all the tasks and I'm going to plot their results so that's just an example of like in a notebook you just

want to like explore the space of different things and then later you might say well we're going to formalize this we're going to make a task we're going to have some parameters from the task what that allows you to do is start to address the evals like with external driver programs so basically I won't get well on this but like once you have this task and this can still be in a notebook and you've got these parameters here I'm

I'm just basically just varying the system prompt and the grading prompt you know I can basically go inspect eval and then I can actually like vary those parameters externally from a driver program or I can do the same thing if it's in a notebook I can say okay I'm still going to keep I'm going to keep my eval in the notebook where I did all my exploratory work but I still want to be able to address it outside of the

notebook okay task variant this is down on the weeds we're not going to get into that okay and then eval Suites

again I'm not going to get into all this but the idea is you should be able to have dozens of evals arranged in directories however you want and we can find them and run them this is an example of like a directory structure that has a bunch of python it has a bunch of tasks we can find the tasks we

can run all the tasks and you know know again the production version of it would probably be more like run it put the put all the logs in an S3 bucket then later on go look at the S3 bucket and retry things that failed and things like that so right okay and then one last piece on sort of workflow is one principle is that if you run an eval from a g repository we want to if you only have

the log file you should be able to completely reproduce the eval it won't necessarily give you all the same OB obviously since the model are non-deterministic it won't give you the same results but you can reproduce all the input parameters and everything so for example if I if if I hand you a log I can use the python API to go read the log I can go get the origin and the commit I can get clone it and then I

can just run eval on it and that will work so the idea is that the the log file is like assuming it was run from a get repo is sort of a unit of of reproducibility okay okay so I think we made it in time to have a decent number of questions but I want to emphasize some resources so one is let me see here is the documentation website lots of documentation that goes into lots of

depth on all the stuff I talked about here there's a fair number of kind of annotated examples you know that go through kind of walk through the code and explain all the different things going on there's also a I can find a benchmarks in git there's like we implemented a bunch of benchmarks and you can see how those are done so lots of examples and lots of docs and then kind of some of the stuff I talked about

workflow and logs and tuning and things is all is all we have docs about that as well and then this is where you would go to get kind of everything I presented so this ha this repo has I won't scroll down yet so people can note the URL I can just stick in the chat let me just do that quickly here or somebody can stick in the chat okay yeah but this basically has

yeah I'll let you I'll let you note that but basically has the slides and then it also has the code so it has if you go into like honeycomb here it actually has the kind of what I actually did the full code and there there's the prompts there's the you'll recognize utils this is like the the code right from right from the course and then we have a here's the query zal you can see as again we reused the code we didn't we

didn't copy and paste it but here's the two eval tasks and then we have a notebook version of that just to demonstrate demonstrate doing it in a notebook so with a little bit more commentary so there's that and then I included some benchmarks here just for so people could explore those and then also that Lang chain example that we talked about is here also so that kind of explains this how to run it and yeah and then the slides so this is a

worthwhile repo to check out and and the docs are also worthwhile to check out so let me go back to here and Go full screen and Q&A yeah so one question is will inspect be integrated with pre-existing posit products like R Studio or anything else so just to clarify inspect is not a posit project it's not it's it's a Uki safety

Institute project so it will not I mean unless it's just it's in a separate exists in a parallel universe so we have vs code a vs code extension but I'm not I'm not sure about about any other other positive things yeah I think I mean I definitely know the answer to this question but I'll just go ahead and ask it because it's the question does inspect support evaluating llm systems

using the inputs and outputs that has produced in the past yes yes you mean you're talking about like some kind of like P logs Trac so what you can do is the input can be I oversimplified input can be a prompt or it can be a message history so you could like replay an entire message history and so you could take like you said a past log and then Rec and then construct a new data set that would allow you to evaluate

using the the previous prompts okay another question is does the inspect team plan to expand it it might be good to maybe briefly sky with what is the inspect team or who's working but does the inspect team plan on expanding the list of metrics to include stuff like Mr Mr and Mr is like a ranking metric yeah we'll we will yes the metrics the built-in metrics are a little a little

spare partly because a lot of people do end up writing their own metrics but but but I think there are ones that are like definitively important and common and so we will definitely be be doing that and then the inspect team is there's two or three of us who work on it working on it full-time but there's a lot of people inside UK AI safety Institute who who provide poll requests and design feedback and things like that it's sort of like just like

eval it's like it's just evals are everywhere and so it's kind of in the air and so there's a lot of feedback and a lot of a lot of iteration then we've got I definitely have the bandwidth to to advance the project like at a significant Pace it's great I guess like this a good time to ask the next question which is what is the future expectation for inspect will it continue to be developed for the long term what will its

direction be dictated by the UK government or the community or both and how do you think that yeah that's a good question it's definitely going to be developed for the long term we view it as like a foundational piece for doing phisticated and complex evaluations and I think I I expect that if it does get picked up more broadly by lots of other players that there will be Community discussion and consensus about what's important and we we definitely

don't it's we would not have open sourced it if it was like oh this is just like something we're using oh by the way here everyone else can use it too I think we want to make it broadly useful for all the different sorts of evaluations what one of the ideas is that if we can kind of level up the the quality of evaluations more broadly I think that's just a better a better world to be in where everybody does

evaluations better and so I think we're we're we're quite interested in making it work for lots of different lots of different use cases and scenarios and and actors okay does inspect allow for using from an API or database query or is it strictly files only so using logs or writing logs I

wonder with what using logs huh yeah I mean the we use like FS speec so logs can come from any file system that is addressable by fspec I think if you had an API where the logs were you would interact with the API you'd realize that on the local file system and then you'd interact with it we have an internal abstraction for logs a log recorder that is not just

doesn't presume that it's a Json file and so we may add other recorders that can you know log to databases and things like that but yeah the the other thing is we we we want the log file format to be something you can you can compute on so we do we actually publish you can see in the docs we publish a Json schema for it we published typescript binding types for it so you know we want it to be something that

people can use and compute on and obviously a python API for it right I'm just scrolling through the questions here someone is really curious about the cyber security eval stuff so question is can you say a little bit more about the docker based cyber security CTF eval do you Envision sharable suets of security tests I think that's going to happen yeah we haven't shared any of ours I do know other people are talking about making sharable

Suites of security tests like there's def other people in the ecosystem who are planning on on open sourcing open sourcing those sorts of things so that's definitely part of the plan and we're gonna we've figured out like people inside UK safety Institute sort of like figured out how to do the docker thing without they just bootstrapped it inside the existing solver idea but we're actually going to have a more formal construct to support what we call

tool execution environments that will you know will do more of the docker heavy lifting and interacting with Doctor compos and things like that that'll be more built into the framework in the future in the not too distant future like if you went to do it now you might say huh what am I supposed to do but I think in month or two it'll be more clear and documented and and like happy path see my team is using weights and

biases to track eval metrics is there a way way to combine inspect AI with weights and biases that you know we are hoping to do that that's on the Fairly short list of things that we want to do so I haven't looked carefully at the weights and biases API and what like how rich we can make it and hopefully we could make it quite Rich I know that they have some kind of API saw where you can like have

an iframe in their thing so we could potentially even have the log viewer go embedded in weights and bies then also project a bunch of the log file results into into the standard weights and buys as affordances so we're going to be looking at that the next couple three four months hopefully someone asked a question there really curious about the design philosophy behind this like where you got the inspiration from they say it's

like very clean in the clarity of thought is impressive it's all Hadley all the time okay I learned a lot from him so

I've seen a lot of his work and so I'm when I'm when I'm when I'm when I'm making when I'm designing a framework I'm like he's like o he's he didn't provide direct feedback on this but he's like the virtual sitting on my shoulder you know kind of keeping me accountable to keeping things clean and simple and

straightforward and composable so that's great one more person is asking can inspect be used with llm proxies like light llm I don't see why not but absolutely yeah okay yeah yeah someone's asking about do you have any Imports plugins for using Mac gpus locally with inspect AI so you can use so yes so you can use whatever oyama

is doing which runs on the Mac I'm sure that they're using metal and things to make inference reasonable I'm I'm not positive but like I can't imagine that that a project like that would wouldn't be using Mac gpus so oyama's one way and then you can with hugging face use the MPS back end we do we do have support for that so that I feel like the oyama has done a better job like reducing the memory

requirements maybe than I mean depends on the hugging face model but we found a lot of like the the hugging face models that you want to evaluate you know you definitely need to have like pretty reasonable GPU GPU or gpus so and I don't know how generally how good like P Tor MPS is I I I don't know like how good it is so I think those that's kind of maybe a good stopping

point you know there certainly other questions but I think we hit the most important ones okay as far as I can see okay teric what do you recommend for people like trying to follow your work and like how to keep in touch with you or appraised of what you're working on or yeah I'm not I do have I have Twitter presence but I don't I'm not like posting on there all the time so that's not a great place GitHub is perfectly

good place to to stalk and see and see what's going on you know some of these commits don't show up like for inspect don't show up there but but some of this like peripheral work like this Workshop show up there so yeah I'd say follow follow me on GitHub is a good way to go okay great yeah all right with that you know thank you JJ it's really great to have you here I learned a lot about the

framework as well so it was great to have this overview all right I yes it's a privilege to be able to to come and talk to to everyone here and so hopefully I'll have for those that decide to give inspect a try hopefully you'll have you'll have success with it and let us know if you don't and you know we'll be very active on GitHub issues so please let us know what's wanting and or what you aspire to do

that you can't do all right great thank you thanks JJ