RAG (evaluate intermediate steps) | LangSmith Evaluations - Part 16

7 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=YX3JMAANGGQ
https://www.youtube.com/watch?v=yx3JMAaNggQ

SUMMARY

Lance discusses an efficient method for evaluating retrieval-augmented generation (RAG) pipelines by extracting intermediate components, such as retrieved documents, directly from the evaluation trace. This approach simplifies the evaluation process by eliminating the need for the pipeline to output all necessary components explicitly, making it more practical for real-world applications.

- *Introduces a new method for evaluating RAG pipelines by accessing intermediate components from the trace.*
- *Highlights the limitations of previous evaluation methods that required explicit output of documents.*
- *Demonstrates how to isolate and retrieve documents using the evaluation code.*
- *Explains the process of grading relevance and hallucination using binary scoring.*
- *Shows how to set up grading prompts and structured outputs for evaluation.*
- *Emphasizes the convenience of this method for realistic RAG applications.*
- *Concludes that this technique allows for more streamlined evaluations without needing to modify the pipeline structure.*

hi this is Lance from lanch this is part 16 our Langs withth evaluation Series so we've been talking about rag evaluation and we covered a few different ways to do it we talked about evaluating the answer relative to a reference we talked about hallucination grading of the answer we talked about retrieval grading of the relevant documents now there's one thing that you might have noticed that I don't like about at least two of our approaches here so for the

hallucination grading and retrieval grading we had to do something that's kind of ugly if you look at a rag pipeline remember we returned answer and the documents both from a rag Pipeline and then with both of those things returned we're able to perform evaluations but this is kind of unrealistic right most rag pipelines will only return for example the answer and maybe some other ancillary information but it's rare that you'll also return the retriev documents or at least you don't want to enforce

that requirement to do evals so I'm going to show a trick that makes this lot easier and what I'm going to show is the ability to actually just in our valuation code reach into our trace and grab certain components not notably for example the documents from the trace itself so I'm going to kind of have this side by side so here's our rag chain like we've talked about before and here is some updated evaluation code I'm

going to show over here and what I'm going to show is here's an update stated document relevance greater we

did this previously but here's a different way to do it and you're going to see something kind of interesting this code up here allows me to basically reach into the trace and fish out the retrieve documents so if you look at our chain there is a function retrieve documents that gets the documents so why can't I you know

reach in and grab those and just isolate those we can do that right so you can see all you do is this is my eval code so this run remember there's run and example so those are the two objects that are passed into our evaluator we talked about that a lot before so from this run I can just isolate child runs and you see I call this get answer now each of my functions that's decorated with traceable then is basically

retrievable using this name so the function name here maps to the Run name over here so when I basically call this child runs from My overall run I can isolate for example all runs name get answer so that's going to basically pull this function here right so I get my rag pipeline run I'm going to call that now I can then call this child runs from

that object and if you think about it inside get answer I do call a few other functions including retrieve docs as we see right here so I can B basically just say okay now get me the run retrieve docs from the children of this initial run that is then going to get the output of this function right here and you can see just I then call retriever run. outputs I get the output and those are just the documents and I basically am

extracting the text from them and in addition I'm also going to get my question so now I have what I need I've been able to fish out the documents for my Trace I have my question and the rest of this is actually pretty simple stuff we've kind of talked about before I'm going to find a data model for my grader so in this case I'm just going to say hey you'll give me a binary score for

relevance of retrieve documents the binary this is a pantic object so you know give me a binary score of one or zero I'll use dd35 turbo as my greater I going to use structured output using this grade document schema and then here's my grer prompt we saw stuff like this before you know if they're relevant pick one otherwise zero right grade prompt I invoke my retrieval grader with question and document text again I set my chain up here using

grade prompt and Plum that to my llm bound to my structured object here and then I just return the score as an INT of course now this should already be an in because the pantic right here but I can again just ensure that's the case and that's really all I need to do so same will apply for hallucination grading it's basically the same idea I can get my documents just like we did before and this this case I just get the generation

from rag pipeline run. outputs in this case I took the in this case it was retriever run inputs. question so that's like the input to my function in this case I'm grabbing the output of my rag pipeline which is the generation again setting a data model in this case great hallucination same kind of idea different greater prompt same flow though my hallucination grader will return one or zero just like we saw before and again I return my score

and that's really it and I have run all that already but I can just show you kick that off again so all I've done is Define these two functions they then passes evaluators it's nice and simple and that's running right now so I can go over to my data set and I'm going run this on rag test LCL so I can show you over here I'm in my data set and here's a run that I just ran

and what I can see here is that just like we saw before show I'm showing feedback from all scores my grades for both are shown here I can zoom into each one if I want to so I can actually investigate the grader itself and I can confirm that and you know this is actually the grer logic that we just defined in that function I can confirm that it's it's reasonable that's good good I can go

back and again we can see this is now really simple so I've been able to evaluate both answer hallucinations and document relevance in one simple step and I've used nice ability to fish out intermediate things from my Trace in this case documents and I just use them in my grader directly so this is really convenient because I don't need to define a pipeline or chain that outputs them and then use them independently I can just reach into my trace and get

whatever I want and use that as part of my evaluator so it's a really useful trick and it's a more realistic way that you would do some of these types of evaluations in particular like we had talked about before things like document grading or an solution grading that require this intermediate this these intermediate documents it's much more convenient to actually just reach into the trace and grab them rather than having the requirement to Output them from my chain which is

unrealistic in many kind of rag applications so again this is a very useful trick for doing evals in particular for rag where you can just reach into traces and get intermediate objects you need for the evaluation thanks