How to Use LangSmith to Achieve a 30% Accuracy Improvement with No Prompt Engineering

15 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=THZTQ_PJSGO
https://www.youtube.com/watch?v=tHZtq_pJSGo

SUMMARY

Harrison from LangChain discusses how the company improved application performance by 30% using LangSmith, a tool designed for logging, tracing, and enhancing data workflows. The tutorial walks through setting up a classification task, collecting feedback, and automating data handling to refine model outputs without needing extensive prompt engineering.

- *LangSmith is a platform that enhances application performance through integrated logging, tracing, and feedback mechanisms.*
- *Dosu, a code engineering teammate, achieved a 30% performance improvement in a classification task using LangSmith.*
- *The tutorial demonstrates setting up environment variables and logging data for a classification application using OpenAI directly.*
- *Feedback can be collected and associated with specific runs to improve model accuracy over time.*
- *Automation rules can be established in LangSmith to categorize positive and negative feedback into datasets.*
- *The application can utilize few-shot learning by pulling relevant examples from the dataset to enhance classification accuracy.*
- *Semantic search can be implemented to select the most relevant examples for new inputs, optimizing the number of examples used for predictions.*
- *The approach can be applied to more complex tasks beyond simple classification, showcasing the versatility of LangSmith.*

hi y this is Harrison from Lang chain today we released a Blog about how dosu a code engineering teammate improved some of their application performance by 30% without any prompt engineering and it's using a lot of the tools that we've built at laying chain over the past few months and so today in this video I want to walk through roughly how they did that and walk through a tutorial that will teach you how you can do it on your

application as well so specifically what they used was Lang Smith and so Lang Smith is our separate platform it's separate from Lang chain the open source it actually works with and without Lang chain and actually dosu doesn't use linkchain but they use Lang Smith and what lsmith is is a combination of things that can be aimed at improving the data fly whe of your application so this generally consists of a few things this generally consists of logging and

tracing all the data that goes through your applications testing and valuation and Lance is doing a whole great series on that right now there's a promp hub there's some human annotation cues but the real power of Lang Smith comes from the fact that these aren't all separate things these are all together in one platform and so you can set up a really nice flywheel of data to to start improving the performance of your application so let's see what exactly

that means there's a tutorial that we put together that walks through in similar steps some of the same things that dosu did to achieve a 30% increase and the task that they did it for was classification which is a relatively simple task by llm standards but let's take a look at what exactly it involves so we're just going to walk through the tutorial the first thing we're going to do is set up some environment variables here these this is how we're going to

log data to our laying Smith project I'm going to call it classifier demo set that up let me let me restart my kernel clear all previous ones now set that up awesome so this is the simple application that mimics some of what dosu did so if we take a look at it we can see that we're using open AI we're not even using Lang chain we're just using open AI client directly and we're basically doing a classification

task we've got this like F string prompt template thing that's class classify the type of the issue as one of the following topics we've got the topics up here bug Improvement new feature documentation or integration we then put the issue text and and then we really just wrap this in the Langan Smith traceable this just will Trace things nicely to Lang Smith and this is our this is our application if we try it out we can see

that it does some classification steps so if I paste in this issue fix bug in lell I would expect this to be classified as a bug and we can see indeed that it is and if I if I do something else like let's do H like fix bug in documentation so this is slightly trickier because it touches on two concepts it touches on bug and it touches on documentation now in the Linkin repo we would want this to be

classified as a documentation related issue but we can see that off the bat our prompt template classifies it as a bug adding even more complexity in here the fact that we want it classified as documentation is something that's maybe a little bit unique to us if if pantic or some other project was doing this maybe they would want this to be classified as a bug and so Devon at dosu has a has a really hard job of of trying

to build something that'll work for both us and pantic and part of the the way that he's able to do that is by starting to incorporate some feedback from us as and users into his applic so one of the things that you can do in laying Smith is leave feedback associated with runs so for this first run that gets a positive score so if we if we run this again notice one of the things that we're

doing is we're passing in this run ID and so this run ID is basically a uu ID that we're passing in the reason that we're creating it up front is so that we can associate feedback with it over for time so if we run this and then if we create our L Smith client and if we create the feedback associated with this this is a pretty good one so we can assume that that it's been marked as good we've collected

this in some way if you're using like the GitHub interface that might be you know they they don't change the label they think it's good and so we'll mark this as user score one and we're using the run ID that we create above and pass in so we're using this to collect feedback now we've got this followup fix bugging documentation it creates the wrong kind of like label we can leave feedback on that as well so we can now call this create

feedback function and notably we're leaving a correction so so this key can be anything I'm just calling it correction to line up but then instead of passing in score as we do up here I'm passing in this correction value and this corre C value is something that's a first first class citizen in lsmith to denote the corrected values of what a run should be and so this should be documentation and let's assume that I've gotten this feedback somehow maybe as an

end user I correct the label in in GitHub to have it say documentation instead of bug so let's log that to link Smith okay awesome so this is generally like what I set up in my code I now need to do a few things in Lang Smith in order to take advantage of this data flywheel so let's switch over to link Smith I can see I've got this classifier demo project if I click in I can see the

runs that I just ran if I click into a given run I can see the inputs I can see the output I can click into feedback and I can see any feedback so here I can see correction and I can see the correction of documentation if I go to the Run below I can see that I've got a score of one because this is the input that was fixed bug and lell and output of that okay awesome so I have this data in here

I've got the feedback in here let's start to set up some Automation and what I'm going to want to do is I'm going to want to move data that has feedback associated with it into a data set so I'm going to do that by I'm I'm going to click add a rule I'm going to call this posit positive feedback I'm going to say sampling rate of one I'm going to add a filter I'm going to add a filter of where feedback is is

user score is one and I can see that actually actually let me switch out my view so I can see one thing I can one thing that's nice to do is just preview what the filters that you add to the rule are actually going to do so I can do that here I can go filter feedback user score one and I can see that when applied this applies to one run so I can basically preview my

filters here I can now click add rule it remembers that filter let's call this positive feedback and if I get this positive feedback I just want to add it to a data set so I just want to add it to a data set let me create a new one let me name it classifier demo it's going to be a key value data set which basically just means it's going to be dictionaries in dictionaries out and let me create

this and I've now got this rule I am not going to click use Corrections here because remember this is the positive feedback that I'm collecting okay great let's save that now let's add another rule let's go back here let's remove this filter and let's add another filter which is instead when it has Corrections so now I'm saying anytime there's corrections I can see the filter applied again go here add rule I can now let's call it

negative feedback I'm going to add it to a data set let's call it classifier demo and now I'm going to click use Corrections cuz now when this gets added to the data set I want it to basically use the corrections instead of the True Value so let's save this and now I've got two rules awesome okay so now I've got these rules set up these rules only apply to data points and feedback that are logged after they are set up so let's go in

here and we basically need to rerun and and have these same data points in there so that the rule rules can pick them up so let's run this this is the one with positive feedback so let's leave that correction let's rerun this this is the one with negative feedback so let's leave that correction and now basically we need to wait for the rules to trigger by default they run every 5 minutes we can see now that it

is 11:58 just 1159 and so this will trigger in about a minute so I'm going to pause the video and wait for that to trigger all right so we're back it's just after noon which means the rules should have run the way I can see if this happened by the way I can click on rules I can go see logs so I can see logs and I can see that there was one rule or there was one run that was

triggered by this rule I can go to the other one I can see again there was one run that was triggered by this Rule and so basically that's how I can tell if these rules were run and what data points they were run over so now that they've been run I can go to my data sets and testing I can search for classify demo I can look in and I can see that I have two examples I have this fixed bug

in lell with the output of bug and so this is great this is just the the original output and then I also have this other one fix bug and documentation with this new output of documentation and this is the corrected value so we can see that what I'm doing is I'm building up this data set of correct values and then basically what I'm going to do is I'm going to use those data points in my application to improve its

performance so let's see how to do do that and so we can go back to this nice little guide we've got it walks through the automations here and now we've got some new code for our application so let's pull it down and let's take a look at what's going on so we've got the Langs Smith client and we're going to need this for our application because now we're pulling down these these examples in the data set I've got this

little function this little function is going to take in examples and it's basically going to create a string that I'm going to put into the prompt so it's basically going to create a string that's just alternating inputs and then outputs super easy and that's that's honestly most of the new code this is all same code as before here we Chang The Prompt template so we add these two lines here here are some examples and then a placeholder for

examples okay and we'll see how we use that later on and now what we're doing is inside this function we're pulling down all the examples that are part of this classifier demo project so I'm listing examples that belong to the this data set and then by default it returns an iterator so I'm calling list on it to get a concrete list I'm passing that list into my a function that I defined above create example string and then I'm formatting

The Prompt by passing in the examples variable to be this example string all right so let's now try this out with the same input as before so if we scroll up and we take this same input fix bug and documentation and if we run it through this new method we can see that we get back documentation notice here that the input is the same as before so it's just learning that if it has the exact same

input then it should output the same output the thing that we're doing by using this as a few shot example is it can also generalize to other inputs so if we change this to like address bug in documentation we can see that that's still classified as documentation there's still these conflicting kind of like bug and documentation ideas but it's learning from the example and it's learning that there should be documentation let's see what some other okay so now you know like does

this fix all issues no let's let's try out some things like make Improvement in documentation is this going to be classified as a Improvement or as documentation so it's classified as Improvement we probably want it to be classified as documentation so one thing we can do is we can leave more feedback for it and so this this imitates exactly what would happen in real life in GitHub issues like you keep on seeing these new types of questions that come

in that aren't exactly the same as previous inputs because obviously they're not and then you can start to capture that as feedback and use them as examples to improve your application so we can create more feedback for for this run like hey we want this to be about documentation great so that's a little bit about how we can start to capture these examples use them as few shot examples have the model learn from previous patterns about what it's about

what it's seen the last cool thing that dosu did that I'm not going to walk through or I'm not going to replicate it in code but I'll walk through it is they basically did a semantic search over examples and so what is this and why did they do this first they did this because they were getting a lot of feedback so they had hundreds of data points of good and corrected feedback that they they were logging to lsmith and so at some

point it becomes too much to pass in hundreds or thousands of examples so rather what they wanted to do was they only wanted to pass in like five or 10 examples but they didn't want to just pass in five or 10 random examples what they wanted to do was pass in the examples that were most similar to the current input and so the rationale there is that if you look for examples that are similar to the input the outputs

should also be similarish or there should be like the logic that applies to those inputs should be similar to the logic that applies to the new input so basically what they did was they they took all the examples they created embeddings for all of them they then took the incoming kind of they they took the incoming input created embeddings for that as well and then basically found the examples that were most similar to that and so this is a

really cool way to have thousands of examples but still only use five or 10 for your application for a given point in time hopefully this is a nice overview of how you can start to really build the feedback loop you can capture feedback associated with runs and store those in link Smith you can set up automations to move those runs and

sometimes their feedback as well to create data sets of good examples and you can then pull

those examples down into your application and use that to improve the performance going forward doing this with classification is a relatively simple example however there are lots more complex examples that we think these same exact Concepts can be relevant for and so we're very excited to try those out if you have any questions or if you want to explore this more please get in touch we' love to help