Benchmarking Question/Answering Over CSV Data

29 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=JGNF4OHPTBA
https://www.youtube.com/watch?v=jGnf4OhptbA

SUMMARY

The video discusses a new approach to improving question answering over CSV data, specifically using the Titanic dataset. The presenter shares insights from a recent blog post that details their methodology, challenges faced, and solutions developed during this process, emphasizing the importance of data collection, debugging, and evaluation in building effective language model applications.

- *The motivation for the project stems from a demand for better question answering over tabular data, particularly CSV files.*
- *Initial challenges included a lack of evaluation data, leading to the creation of a simple application to gather real user questions and feedback.*
- *The team utilized Langsmith for debugging and logging, which helped identify areas for improvement in their question-answering model.*
- *They discovered that allowing the model to execute arbitrary Python code improved its ability to answer a wider variety of questions compared to using a constrained set of functions.*
- *Key insights included the importance of data formatting for effective model responses and the need for robust evaluation methods to assess model performance.*
- *The final solution involved a custom agent that combined a retriever tool and a Python execution tool, improving the model's accuracy in answering questions.*
- *The evaluation process highlighted the challenges of semantic equivalence in answers, prompting the use of language models for assessing model outputs.*
- *The project aims to contribute to ongoing improvements in question answering systems and invites community collaboration for further enhancements.*

hello everyone Thanks for tuning in this is something new we're trying it's going to be a video accompanying a Blog that's a deep dive on a particular topic it's something we haven't done before so if you like it leave let us know in the comments if you don't like it also let us know in the comments this will just be one take raw unedited because I don't own any Fancy video or audio editing equipment and basically what we're going

to do is we're going to walk through this deep dive blog that we recently published on our website where we duped Dove on basically benchmarking question answering over CSV data and the so what we're going to cover here is basically a bit of background motivation and I'll include links to this blog and all the relevant things in in the YouTube description but what we're going to come into here is background motivation why this is an interesting problem why we're doing this

then we're going to talk about how we got started we really started from a place of where we didn't have much and so we need to get started somewhere and so we'll talk about how we got started and why we got started there then we'll talk about using langsmith to kind of debug what's going on the end goal of this is to produce better question answering over CSV data and so we'll talk about some of the debugging

that we did we'll talk a bit about our evaluation setup so we measured a few different Solutions here and then finally we'll talk about our final solution at this point we we put in maybe about two or three days total to this so the the final solution can almost surely be improved and hopefully some of the advice some of the ideas here spark you guys to go out and improve it and would love to see the

improved Solutions so yeah as a basic background motivation we've done a lot of question answering over your data in link chain that's been one of the core things from the beginning for unstructured data like text there's a pretty standard recipe where basically you take the data you split it into chunks you create embeddings you put it in a vector store and then you do some retrieval on top of that and and that's pretty standard and we've we've been doing that for a while

one of the areas where we've gotten a few comments for basically people wanting better Solutions is question answering over tabular or CSV data in this case and you know we'll there's other types of tabular data there's there's data in SQL tables we're actually working on something for that as well we'll treat those as like slightly separate although obviously they they're very related but but one of the biggest areas that we've always got asked to

improve is question answer or CSV data and so we wanted to do that we wanted to get started doing that there's a lot of valuable data in CSV we wanted to try to figure out a best two question answering over it so that's the that's the basic motivation for why this problem and then why we're writing a blog post about it is that you know we think this is pretty representative of a lot of tasks that

people will do when they're trying to improve their own LM applications there are a lot of challenges here the main ones being we didn't really know how to even measure if we were improving it and so a lot of the the time that we spent doing things was on Gathering a data set to evaluate and then figuring out how to evaluate and then the the other reason we think this is interesting is a lot of the

insights that we gathered are they're not anything super revolutionary a lot of them come down to like good data processing and and debugging that and you know that's that is one of the big issues people face when building all our applications so we also just wanted to shine a little bit of light on that and how we were how we think about doing that how we did that what tools we used stuff like that okay so that's the basic background on why

question answering over CSV data and why we even wrote a blog post about this in the first place so as mentioned we need to get started somewhere we didn't really have any data to evaluate on and this is a con this is a problem that we think is pretty common in all applications unlike in traditional ML and data science you

don't usually start with a data set because these alarms are fantastic zero shot Learners so you can basically you

know start with an idea tell it what to do and then you have an application and that's really good because you can get an application super quickly but that's not that great for evaluating because you don't have any data so the first thing we want to do is come up with some data to even use as an evaluation data set and we kind of had two ideas one we could come up with a bunch of questions

ourselves we could generate some maybe programmatically the benefits of that where we could probably do it you know we could do it relatively quickly especially if we did it programmatically the downsides of it were that we'd be biased in what we thought the questions people would ask of a CSV might be and we wouldn't actually get any real data on what types of questions people would actually ask in in practice and so we went for a different solution

which is basically we wanted to gather these these real questions in the wild so we set up a super simple application we fixed the CSV we we did a we did a question answering over the Titanic data set the Titanic data set is a fantastic starter data science data sets used in lots of getting I think it's the getting started data set for cargo competitions and we fixed that as the data set we put out an application where people

could ask questions of this data and then basically in the background we logged everything to linksmith our our logging debugging testing evaluation platform and then we also hooked up a little button into the UI with a thumbs up thumbs down and and then we asked people to to leave feedback and so what this looks like is something super simple like this we use streamlit streamless awesome it was really easy to get started who was in Cabin

ce128 we can ask a question we can press submit it should hopefully run in the background we'll get back an answer and then we can leave feedback here and so this feedback was very valuable in coming up with our data set because let me show you what I did so basically this is the project in langsmith where everything was logged and so you can see all these different runs

in that that happened from the app and and the code for the app is open source so you can easily kind of see what's going on this is the run that we just ran I believe yeah who is in Cabin c128 and then we can see that it has a feedback of one logged here filtering on this sidebar here we can now filter data points that have good scores or bad scores

and basically narrow in on on data points that have some sort of label which we can then use to add to a data set and so I guess there's two parts here one why would we want to to use why why is this helpful when constructing a data set this kind of gives us an idea of where the model might be messing up so users have very nicely kind of like highlighted where responses were bad or even

more accurately where they thought responses were bad sometimes the responses are okay they just they

weren't satisfied with the the realism of what this bot was intending to answer and so we can drill in on these data points which we think are bad and these are really really valuable because they represent cases where our current solution is not performing well and so if we want to improve it the first step's probably improving it there so now that we have these data points we

can click in to anyone and so we can see this we can see the trace of what happened and so here we have this agent the agent didn't actually do anything it just responded from from the open AI model and so we can see that we have this input and then we can see this have this output and basically it looks like here the agent didn't actually do anything under the hood it just kind of said some some

python code that you could run on the CSV but it didn't actually do it and so that's why the person probably gave it this this feedback of the zero so now that we have this data point what we can do is we can click add to data set this then presents the inputs and outputs here here we probably want to change the output I don't actually know the number of cabins that there were on the Titanic when I was doing this in

for real I had a separate screen open on the side with the CSV and I was running I was looking at the CSV running some some python code to basically answer these questions for real and then writing the answer let's say the answer is like 42 or something like that I can then choose the data set that I want and click add and I'm not going to submit this because it's probably a wrong answer but basically that's the workflow

that we did to to gather a data set of examples based on the the CSV application that we had here so thank you to everyone who who tried this out and left feedback that was that was very helpful we use that to gather 50 examples and so you can see the examples here this is this is open source at one of the or we we downloaded this

into into CSV format and it's in one of the githubs so I'm not going to go into too much detail here but basically you can see kind of like the inputs and outputs we gathered about 50 different examples this way along the way we also got a little bit of sense of what was going on and specifically why some of the examples were bad and and I talk about this a bunch more in the blog post so I

might I'm maybe going to skim over it a little bit here but basically there was two big areas where we where we found some room for improvement and to understand those two areas let's maybe first talk about our initial solution we figured right off the bat that new question answering over CSV data there would be two different components that were needed one would be some type of basically way of running queries against the data frame and and so let's let's

call that basically ways of running pandas or python code to manipulate the pandas data frame that's needed for answering a lot of questions like how many girls and boys survived so you need to do some sort of filtering how many rows are there things like that at the same time we also thought and this turned out to be pretty true that there's probably a need for another action or another tool

that that an agent could have which is basically looking things up and doing doing kind of like a retrieval step and this is really important because of the some of the data in the CSV is a bit more textual in nature specifically the the people's names and so you can have questions that reference a person's name and it can be a little bit tricky to find the like you wouldn't want to do an exact match based on that you might not

even be able to do like a stir contains on that if you're if you're running python code and so it's simpler to have another tool where you can do some sort of like semantic lookup based on some sort of query and so the initial solution that we had had a retriever that basically we indexed the each row as its own document created a retriever of those documents that was one tool the other tool that it had was basically a way of

running python code and we we had two choices here to start one we could have used cork cork is a library written by Eugene who's who's one of the people who works at langchain it's basically a way of letting the language model kind of have access to specific functions that it can cause so it doesn't write kind of like it doesn't just like write Python and then execute python it writes kind of it writes a

program in in this kind of like pseudo language called cork and and then kind of like executes that written code language more or less the benefits of this are that you're not executing arbitrary python so it's really safe the downsides are that you have to explicitly list out all the functions that it has access to so to start we gave it access to I think four or five different functions there were things like calculate like the average of a

column sort a column see the top five values of a column and and you could execute these things in one after another as well so you can do things like short this column in ascending order and then give me the top five columns and so much more constrained than executing arbitrary python code and that was one of the things we realized was a bit of a downside people were asking a ton of questions and the five functions that we

gave it weren't nearly enough to cover the wide tale of questions that people were asking and so you know that was a learning from for us putting it out into the world is basically there's a lot of different variants of questions that people can ask if we did and skipping ahead a little bit you know in their final solution we we kind of gave it access to Just Right arbitrary python code to manipulate the pandas data frame so it could use all of the

pandas built-ins that it knows from being trained on another thing we could have done is now that we had this big Bank of questions we could have come up with more and more functions and given Quark access to those and tried to fix it that way but we opted for the slightly easier way of just letting that kind of like write what it knows which is Panda's code the other the other big issue that we discovered

is a really interesting and seemingly well it's seemingly boring it's seemingly kind of silly bug but I think it's pretty representative of a lot of it's pretty representative a lot of the big issues in actual difficulties of building LM applications and I think I have a trace here yeah okay so you can see so this is so this is one of the ones that we got some bad feedback on and and

it's actually the question that we had in in the first place which is who in cabin c128 and so we can now see kind of like the execution we can see we jump in we get back this call where we filter kind of like cabin cabin equals this this looks pretty reasonable but when we look at the output of the Python Rebel we can see that it's actually cut off a little bit in the middle and so then when we go to answer

let's Zoom back out here the name of the passenger isn't displayed because that's one of the columns that's cut out this is pretty subtle but what is basically going on is that pandas has Max columns to display parameter and it actually was different on my local computer compared to the hosted streamlit app and for whatever reason the environment that I was running on streamline had some default that was a lot smaller than on on my local

computer and so when I was testing it out on my local computer it's working fine but then when I pushed it to streamlit it was it was it was not working fine it was pretty bad and so this seems like kind of like a random a random silly bug but as I said I think it's pretty representative of a lot of the issues that we see people running into and spending time debugging in llm applications which is basically there's

there's two there's two big areas of time that we see people or two big areas of work that we see people spending time on one is on prompt engineering and I think everyone kind of like knows what prompt engineering is and why it's important and why it takes a lot of time and is annoying so I'm not going to go into that here but another area that we see a lot of people spending time on is

basically data engineering and getting kind of like the data to go into the prompt in a way where the language model can make sense of it and this can involve a few different things this can involve kind of like setting up pipelines so the correct data that takes a lot of time this can also involve formatting that data in the the right way and so this is an example of where the data when it went into the language

model was not at all formatted in a way that set the language model up for Success when trying to answer we've seen a lot of we've seen a lot of bugs in a lot of kind of like instances of this where because you've got this sequence of calls that are sometimes deeply nested where you've got this flow of data through various layers by the time it actually goes in the language model it may not look as you

expect it to and so being able to see exactly what went into the language model as well as exactly what came out of the language model and exactly what came out of each tool is really really valuable for that reason and so in and so the fix here was just simply changing kind of like the max columns to display from pandas you know for for the bugs that we've seen of this nature before all of them have their own

kind of like unique solution there's no kind of like one standard thing like data pre-processing matters a lot when you're trying to give the language model something intelligible to reason over so those are some of the insights that we gathered from from doing this and I guess now I'll talk about so and and all this is in this repo that we put up link chain benchmarks we are going to add more things in here right now it's just question

answering over csvs we're going to add one for SQL that's coming soon we're going to do a lot more after that and there's a bunch of different things in here there's the streamlit app so the code for the streamlit app so that you could run it if you want there's data.csv so we've put the 50 different examples that we gathered the the questions and answers in here and then there's also code for evaluation so talking a little bit about

evaluation if we look at some of the questions you know the we have ground truth answers but the issue is there can be a lot of different ways of saying this that are still correct and so the challenge is gauging whether what the language model says is actually equivalent semantically to to the outputs here and so that's what makes evaluation hard and so and so in order to do that we're going to use a language model itself

and so basically we're going to do everything through langsmith we're going to upload our data set there we're going to then run test runs against a bunch of different agents or a bunch of different solutions I should say because some of these aren't agents and where we're going to Benchmark them these are screenshots taken from linksmith I'll show this in langsmith after that or after this but just talking really quickly about a few of the different solutions that we

Benchmark we have a pandas agent in link chain so we Benchmark that with GPT 3.5 gpd4 Benchmark pandas AI this is a really cool package that's explicitly focused on question answering over Panda's data frames so we benchmarked that and then we bench and then we came up with a final solution which is a custom agent and so I'm going to jump into that and then I'll jump into the evaluation results but basically this is the final custom agent that we

came up with first thing we did we set the Panda's display option of Max rows and Max columns to avoid any type of funniness in how the data is represented this is going to be an agent so there's a few things to think about when creating the agent there's the props that you're using there's the the way that you're prompting the agent to choose a tool and then there's tools that you're giving it so the first tool that we're going to

give it is this retriever tool so we're creating where I guess we we load the vector store here because we created it in a separate script I think we have the the script for that here as well and so we're creating this Vector store we're loading that and we're creating this retriever tool and so we're giving the agent a tool that take that that is a retriever under the hood so it takes in a query and it returns a list of

documents and then we're just passing those documents in into the context and the agent can answer it this here is now the prompt that we're using for the agent and so you can see that we tell it that it's working with the pandas data frame we then substitute in the head of the data frame this is pretty important because we want the agent to know which columns it has available and what the shapes of those columns are and what the types are and

things like that so we give it the head of the data frame there we enclose it in in some XML tags here this is I found this pretty useful for basically denoting different sections and then we also tell it that it has access to this other tool which because the only column in the Titanic data set is is the name or the only textual column in the

Titanic data set is the name we're calling it explicitly that and then

we're providing like a few examples on how it should answer specific questions so basically telling it when it should use one tool versus the other I had to add this in because it was getting a little bit confused about when to use the the vector store tool or the retriever tool let's go down we're now formatting this template we're now creating this chat prompt template

we're creating the second tool this is the this is the python reple tool this is pretty standard and then we're using the openai functions agent and basically this is an agent that just uses the functions calling ability of open AI to decide which tool to take where and then we're initializing the Asian executor the Asian executor is basically the loop that does the call agent decide what action to take call the tool

pass the result back into the agent repeat until it decides it's done or until it's reached five iterations at which point it will generate a final response we're then defining our evaluator which is this built-in evaluator question answering evaluator and then we're running it on the data set so we're giving it the name of the data set that we uploaded it to this Factory method to get the chain the evaluation con config and then decline this is

the linksmith client what this creates is a bunch or this creates a single test run these are a bunch of test runs because we ran this on a few different agents and so if we click in to a given test run we can see each of these is a data point they all have feedback here we can see some summary stats at the top this is the best agent that was ran it got about 92 percent

correct we can we can end actually one really important thing to note here is that the evaluator that we're using because it's a language model is not perfect and it messes up and we will actually look at some examples where it messes up and I think there's some discussion around whether using language models as evaluators for language models is actually okay or whether there's errors with it whether you should do it at all I think my personal take is it's not perfect but

it's probably the best thing that we've got and I've found it useful for helping to basically draw my eye to points that I should look at so I'm not going to trust this 92 percent exactly I'll probably trust the ballpark of it and then I'm also going to know what data points to look at to see which data points failed basically one way to think of this is kind of like running a bunch of unit tests over all these data points

and seeing which ones passed or failed and so I can do exactly that I can click in again filter on this see which one's failed let's look at this one so this is one where it asks how many people are named Charles it uses the vector store retriever tool unfortunately it's limited to only return for the top four results so it thinks there's only four people named Charles that's good to know maybe we can improve that in the future

let's look at another one this is one where it probably you know the the output is is correct the that the reference output that we have just expresses the result in a different way but these are basically the same so it probably

should be correct unless we really cared about the exact formatting of what we meant by by a ratio and and and then we can also and then

so here's another one please list all survivors with age greater than 50. we can see that it does a pretty good job but it doesn't list all of them it knows that there's 22 total but only lists 10 and if and this is where kind of like having this hooked up to langsmith so you can see the exact Trace is really really handy because now we can see what's going on and we can see that the python Rebel up it gets cut off

in the middle and so when it gets passed into the language model you know it can see you can see the 10 but it can and it knows that there's it looks like it knows that there's 22 total yeah notice that there's 22 total but it doesn't know what the other ones are and so that's why it's responding that way one other thing that we can do with this evaluation bit is basically for all the feedback because the feedback itself is

an llm call we can jump in to the alarm call to debug why it's giving that piece of feedback so here we have this QA eval chain we've got the input of a query an answer and a result we've got the output and then we can jump in and we can see exactly what's going on inside this chain that's calling a language model to get the the evaluation result so here we can see the prompt that we're using

we can see the instructions that are given the format that's given and then and then the output that comes out and so it's nice to have this hooked up as well because when you're using when evaluators are themselves LM calls you also kind of want to be able to inspect the evaluators and debug what's going on there I think that's pretty much everything that we wrote about in the blog there's a lot more details in the blog we tried to

call out some kind of like General stuff or general takeaways you know part of this is specific to the Titanic data set that's not super interesting part of this is pretty General towards evaluating either CSV or tabular question answering applications or llm applications in general and so those are more interesting so we tried to call some of those out really you know we don't we don't think this is perfect we think this is kind of like a step in

the direction of adding more kind of like rigor and benchmarking a pretty common use case that we heard a lot of requests for and you know there's almost surely room for improvement there's room for more data points there's room for better evaluation metrics and there's of course room for better solutions to this problem and doing the question answering so we'd love any and all contributions really hopefully it's pretty easy to either

contribute to this benchmarking repo or link chain itself and that that's pretty much it thank you guys for listening to this if you found this interesting please let us know maybe we'll do more of them if you didn't listen to it at all then that's also great bye-bye