Context Engineering: Lessons Learned from Scaling CoCounsel

20 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=SLFV3RSJ_D8
https://www.youtube.com/watch?v=sLFv3RSj_d8

All right. Um, hey everybody. My name is Jake. We're going to be talking about uh lessons we learned in what we used to call prompt engineering may still in the future call prompt engineering while building a product called co-consel. So uh you might be wondering like who the heck is this guy and why am I listening to him. Very brief background. Um I founded a company called Kstax 12 years ago. Uh summer 13 at YC. So it's been a minute. Uh I am that old in fact. Uh fast forward like 9 years and we were working on AI and law for the entirety of our existence and as a result we got early access to GPT4. We were just very close with open AI before even chat GPT came out we were using GPT4 and what we noticed immediately and I'll get these two bullet points out at the same time.

So we built this we started pivoting the entire company essentially around uh co-ounsel we had a different business it was based in AI it was business in in law but we pivoted the entire company around chat around uh GPT4 because what we saw early on was that GPT4 could finally unlike GPT3 or 3.5 or any other model we've seen or developed ourselves was able to do complex legal tasks at a rate that was not perfect but was also around the same rate that humans can do for a lot of these tasks. And of course, you can scale it, unlike humans, to hundreds of different things, uh, tasks at the same time. And even if you couldn't scale it in certain circumstances, still faster than people.

So, you had something all of a sudden that was faster and better at legal tasks, the area that we served. And that blew us away. were the guys that did the study that showed that GPT3.5 scored in the 10th percentile on the bar examination. But when you did GPT4 on the same exam, it got to the 90th percentile. And this is an exam that was not in the training set on like basically all the uh here we go the eval today. Um and so we were just absolutely blown away with this model and built a whole product and company around it.

like f you know it was pretty massively successful in legal the idea of having the first AI assistant for lawyers um it's something that a lot of our customers were looking for this functionality that we were offering through co-consel it's something that they'd been asking for us for like the first decade of our business and we're like that's ridiculous we can't build that we need some genius AI to do that and then all of a sudden you know kind of handed to us so we're able to to be successful uh in part because we're just doing what our customers asked for and based on that success uh we were you acquired by a company called Thompson Writers uh in 2023. So, literally two days ago was my 2-year anniversary, which when you all do the acquisition thing, you'll find out is a very important milestone. Uh and uh and uh you know, we've continued to hone some of the uh prompt engineering or context engineering techniques since then, which is why we're talking to you about it now. Um,

before I get into the fun and meat stuff, just in order of semantics, I'm actually not sure about context engineering as the meme. It may or may not be the right thing because as I'll talk about, I think that most prompts are instruction plus context. At least most prompts of consequence. Instruction plus context. So to call it context engineering is really talking about in some sense one part of it, but it's just semantics. Let's get into the actual um you know how to do it whatever you want to call it. From a high level there are really three big steps that we took when we're developing coconsil that I think still apply today. The first is um and this is developing the whole application and then we'll like work backwards from there to where each step of the application fits in and then how each prompt or context whatever you want to call it these days uh fits into that. So the first big step big big picture is like what is the experience you're trying to deliver to customers. In our case it was uh a suite of different what we called skills that co-consul could do. So imagine co-consil almost like chat GPT with tools basically and each tool was another skill and those skills could be things like do complex legal research or read over a hundred or a thousand documents and tell me what's in them or review a contract and tell me what needs to change in order to be like what my company needs. And these skills mapped on for us pretty well to what actual lawyer skills would do. These are the kind of skills that if you're a lawyer you might list on a resume of skills like here are the things I can do. I could do research, I could do contract review, etc. That's the way we thought about it's not the only way to think about it. But in our case, it was a chat application with tools. Your case might be a big button you press and then it randomly generates a poem or it might be uh upload a document, analyze a document, something comes out. But there might be different UIs. But but be really mindful about what is like the ideal customer experience. And then from there, for us anyways, we ask how will the world's best lawyer do that task?

So for legal research to use as a concrete example the world's best lawyer with like an infinite amount of time would take the question given to them and maybe first they just clarify. So prompt one clarify what is you know I need to understand is you trying to search just federal case law or state case law are you trying to uh understand this angle or that angle right so there might be a clarification step and in fact you saw this creep out later in deep research one of the iterations from chat GPT if you saw that right so maybe that's one step but after the clarification step then you have a step where you formulate a number of search queries that's what I would do if I was a a real legal researcher as a lawyer as I did before Ksix as a lawyer. So, you know, so I might, you know, try this kind of search query, that kind of search query, the kinds of words, that kind of words, maybe 20 or 30 or 50 different search queries and then I'd execute those search queries and review each of the results one by one. And the best law in the world would read every single page of every single result that came back and analyze, is this actually relevant to what I'm doing? If so, how does it help answer the question? Maybe I'll make some notes about how this case relates uh to the to the final answer.

compile that into a big notepad and then finally I take all those notes and compile it into a final answer I give to my my client. Those are the steps for that that task. Sometimes the tasks will be pretty linear and predictable and in that case don't be agentic behind the scenes. Just just program it in like if you imagine Python code it's like deaf step one def step two step three and then you just like literally just you know do task def one def two def whatever step one step two step three right. Um so sometimes it's like really linear and sometimes it's aentic. It's like actually if you do a number of searches but if you realize you're not finding what you're looking for then go back to step one and try some more searches. Try some other you know right?

So you have to figure out for yourself how would the best person do this and are they acting more agentically so to speak or more just steps. So how would the best person in the world do this? That's your architecture for that skill or task or whatever in my opinion. That's at least version one or version zero. There may be things that humans aren't even able to do that the world's best X could not even comprehensively you know there's no way that they could possibly do it. Then maybe you can be a little creative here. But I'd start with how the world's best human attack attack this problem. And then within that there are multiple micro steps. I told you some of them. Clarify this the search query. Um run number this number of searches. Review the results that came back from the searches. Review um based on those results, you know, start making notes. These are all micro steps and there may be micro steps within that.

Each micro step is either code or prompt. And increasingly in difficult tasks that we're all doing these days that were un impossible to do like unthinkable to do before GPT4 basically but there's now possible to do those are mostly going to be prompts. So that's that's that's how you work down to like the prompt level and just it's important as a developer to understand where in the flow what this what is the the job of each prompt right like what is the job of each prompt doing when it's trying to accomplish this task for the user from there you have to make the prompt actually work and this is where most people try and fail and then move on because they're like this is an impossible problem to solve I'm not going to build an AI application so the the First thing I recommend like the way the method that we we work through and it's pretty simplistic but you'd be surprised about how few people do it this way. So I would recommend even it may sound really stupid and simple that to try to do it this way and step one is you just write a prompt that's your like best guess of of of achieving this activity. So if the activity is write one good legal research query and then you define in the prompt you know all the instructions this is where like con uh uh having some subject matter expertise really helps. So if you're like a really good researcher, you can give it these instructions about how to write a great legal research query. And then you also write start just like 10 emails. What is a good legal research query? And the idea is given the prompt which is the context and in this case the context is the user query and your instructions. What is the objectively correct answer or how would you measure an objectively correct answer? And start writing down these emails.

Here's where tooling gets into play. I when I develop at home, I use Prompt Fu. It's free. It's open source. It uh is is simple and pro and command line. There is a host of tools though that are paid and cloud-based and are awesome. I've used Vellum before. I don't know if they're here. Those guys are cool. There's so many different tools. Just pick one and make sure it can do this task of you write a prompt, you give it con you write instructions, you give it context. Uh context in other circumstances, by the way, might be the full text of a document. You're asking questions about that document. The prompt might be instructions around doing the next chess move. And the context is the chess game up to this point, right? So instructions plus context. Uh whatever it may be for you, make sure that given this the following instructions in context, it actually does the task well. And uh what my guess is is that the first prompt you write, it's going to pass six out of your 10 tests. And that's the moment most people go, "Well, AI is stupid. I guess I wait to GB6, I guess, right? It's over for me. This this application is doomed." And here's where the hard part is. You just keep working until you pass all 10 tests. even if it takes you a week or two weeks. What I tell everybody who works for me is that the definition of a good prompt engineer is somebody who can write pretty well and also like concisely directly um understandably write great instructions and also somebody who's willing to not sleep for two weeks straight

until they get it right. That's what it takes. Because what you'll find is that you thought your instructions were clear, but then it gets it wrong every time on this dumb question. Okay, let's read the instructions again. Where is it unclear?

Where am I not being clear to this AI? Where am I using the wrong model or doing the wrong setting? I temperature to two, which is somehow possible, or 10 is somehow possible with Gemini models. I don't really understand that. It's like crazy. It's like one is like kind of crazy and 10 is like let's go nuts here. But maybe it tries temperature zero or 0.5 or whatever, right? Maybe I you set thinking too much or too little or whatever. You just keep on experimenting mostly with the instruction because these models are very smart and they will follow your instructions better and better and better but with everything else as well model settings etc until you get 10 out of 10 and then you keep on iterating and iterating iterating and then you go to 50 and then 100 and then a thousand per prompt. Okay, this is what it takes because by the time you get to a thousand and another kind of test here is not the thousand that you thought of by yourself just in a dark room, but what are the kinds of things your users are likely to throw at you? Can you anticipate pre-launch what are the kinds of crazy your your users going to pump into it? We were shocked by how dumb the legal research queries people put into coconsel. We were just totally blown away. But that's the reality of our users and many users uh who even people who are highly educated like lawyers. So be prepared to try to anticipate as best as you can what lawyers will lawyers and I said lawyers actually users will put into your uh into your application what kind of the context will look like and maybe have a small beta program where you tell your customers it is going to be bad at first. We're looking for that. We're looking for you to tell us the times you tried it and give us that that information so that we can put it as part of our tests. Right? That's how to get to a thousand tests. Usually driven by your customers.

All right. I kind of dumped on context engineering as the meme because I think it's context plus instructions equals prompt. But again, we'll probably hear some disagreements throughout the presentations. Doesn't matter really. It's just semantics. But I will say that context really, really, really, really matters. And a lot of people overlook this. There are times where we try to go back to this legal research example. We're like, I can't believe the AI is getting the answer wrong. It's so stupid. But then we actually just sat down and read and sometimes it's like 10 pages or 100 pages. Just read it. And we realized, oh, given this information, I would say the same thing the AI does. It doesn't have the right information. This is especially the case if, and I recommend this, you tell the AI, don't answer based on any of your previously existing knowledge, just based on what I'm giving you right here. If this thing says the sky is purple, the sky is damn purple. Right? That's a good instruction to give. By the way, if you're worried that the AI is going to hallucinate based on the information put in, you just be really, really, really clear. It must be based on the context given.

Okay. Well, but what happens if your retrieval sucks and the information that comes in is is garbage? How could you possibly expect a human, let alone an AI, to answer the question accurately if the retrieval sucks? So now you have to you think you're working on prompt engineering or context engineering. What you're really working on is retrieval. Go Chroma. Uh or you know or maybe you uh have another problem. For us, believe it or not, like OCR was a huge issue.

Legal documents are total messes. Like absolute messes. And what we'd see time and time and time again is that when we read the OCR, it was gibberish. It was total mess. No human in their right minds could get the right answer. So sometimes they able to get it right and we were shocked because the words were out of order. it still figured it out. Okay, so now you have an OCR problem, not a prompt engineering problem. So that that like really opening up under the hood and seeing seeing it like the AI sees it. Read it verbatim, including all the information put in the spacing.

Yeah, it sometimes matters. If it's hard for you to read, it'll be hard for the AI to read is the general rule. AI can do some things you can't do, but if you can make it readable and easy, then you're in a really good place. So that's what I was just saying. Look at the look at the actual like actual information in there. And again, I want to double underscore this because it's going to be the difference between you making a prompt that works and a prompt that doesn't work. If you're not willing to stay up or you or your employees or whoever are not willing to stay up for two hour, two weeks straight, not sleeping, just working on the prompt, you're not going to make it. All right?

You just need to do that in order to get to a place where the prompt's actually working at at scale accurately in the way you want it to. So that's that's our secret to success. It's actually pretty simple. Evals per prompt. Given this information and instructions, it should answer this. If it answering this, it is red, not green. I have to keep on working on the instructions or the information that's coming in until this is green. Let's move on to the next test and the next test. The next test um by the time you're passing like 999 out of a thousand and that one is kind of debatable. Now, you're ready to go uh release to a wider audience in my opinion. Right now, now you have some degree of reliability. It's not guaranteed and you should never tell your customers it's going to be perfect because it's not going to be perfect, but it'll be really good, especially if you anticipated all the different edge cases your customers will throw at it.

A few other quick tricks. First of all, the way that the AI works, as you probably know, is it reads over the entire prompt and also by all the tokens it's generated so far for every token has to generate. So for every token you generate, it takes a while. And this is less true now. the bottles are getting faster and they all quantize like crazy and so on. But when we started GBD4 was slow and if you want to do something like read like a million documents over the course of a few hours or even a few minutes. Okay, well good luck if it's generating a lot of content per uh per document. But there's a trick here. Make it just have one token as its response. That means it does that whole process of reading the entire context and outputting a token once as opposed to twice or three times. Right? And there are some really creative things you can do here with stop words. One thing we love to do is and this is more true of completion models and chat models and unfortunately the world's move to chat models but we would like give the whole prompt tell it the output format is like first you just tell us the number on a scale of 1 to 10 of whatever we're trying to look for just the number and those are all just one tokens each 10's also a single token right or a single word like true or false or if there's a gradient there of you know true false maybe whatever right and then give your explanation dear AI here's the craziest the AI will give a slightly better answer because it thinks it's about to give a whole explanation as to defend its number. But you use stop words or max tokens at one.

So the AI just outputs one one token and then you just cut it right off. It's like stop talking, right? You've got your fast response. Uh and you also got the AI to think a little bit harder before it's given its answer. That's like a trick. Um, similarly for eval purposes, even if you're going to have it output its whole explanation, you want a big nice JSON blob that has all the stuff in it, make it give its uh numeric or objective answer first. That makes eval so much easier. Evals are way easier when you can say like matches word x rather than having llm as a judge for every single one of the eval judge is sometimes the only you got. Um, break things into small parts. is tempting to try to get the AI to to in one prompt or within one context to do like dozen steps and sometimes it works and if you can do it but it's just like people uh when if you give them a simple task they'll more likely to do it right more often break it down to simple steps finally something that we're seeing I think finally the thing that we're seeing is that reinforcement fine-tuning is a lot better than older fine tunes a lot of the people in this room are probably turned off to fine-tuning because you tried fine-tuning if you're anything like us and the gains are minimal at best and you go okay well why am I getting thousands of different examples just for it to kind of up um in the same rate it's up before in fact maybe even get dumber which we saw sometimes reinforcement finding on the other hand requires max 50 to 100 different examples of prompt what good answer looks like and how to uh objectively judge whether prompt whe the answer is good which is the hardest part of creating these reinforce and fine-tuning like fine-tunes for for APIs like OpenAI or if you're doing it yourself. But it goes a really long way.

They go a really long way. And we're starting to a place where we have a different model for every single prompt. And there are hundreds if not thousands of prompts at this point throughout the co-consil ecosystem for each little micro step. Each micro step gets its own reinforcement fine tune model. And that's another way that might get your eval to go from 9900 out of a thousand to 999 of a thousand. Finally, try different models for different prompts. There's no rule that says it has to be GPT all the way through if you have seven different steps. Just try different models. Um all these these platforms now including Prompt Fu which is free and on your on your desktop environment lets you say try it under different conditions including different models. And the other thing too is you might save a buck. If you can go with GD5 mini as opposed to GD5, you might save a lot of bucks. Uh Gemini 2.5 flash, beautiful model, cheap as If it works to your level of satisfaction, use that.

Don't use 2.5 Pro. So try just experiment with different models and it might take you pretty far. Especially if you have a potentially lower than 100% accuracy rate, it's still acceptable for your customers depending on the use case. All right. Well, that's all I got. Hopefully it was useful and we'll do questions at the end, I think. [Music]