Multimodal Evals: #34

61 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=JZHVO0IAX_I
https://www.youtube.com/watch?v=jzhVo0iAX_I

SUMMARY

The podcast discusses the importance of understanding data and system design in developing reliable AI models, particularly in the context of processing receipt data. The conversation highlights the iterative process of refining prompts and data models to improve extraction accuracy, emphasizing the value of real-world data and the need for effective evaluation systems.

- *Learning to code and understanding system design is crucial for building effective AI solutions.*
- *Real-world data, such as receipts, often contains complexities that require careful analysis and tailored solutions.*
- *The transition from using simpler models to more advanced ones, like Gemini Flash, can significantly enhance performance.*
- *Building a robust evaluation system is essential for identifying and addressing errors in AI outputs.*
- *Iteration and experimentation with prompts and data structures are key to improving model accuracy.*
- *The podcast stresses that while off-the-shelf evaluation solutions exist, custom-designed systems tailored to specific problems are often more effective.*
- *Human-in-the-loop approaches can enhance AI systems by allowing users to validate and correct outputs, improving overall accuracy.*
- *The discussion underscores the importance of understanding the problem space and leveraging insights from data to drive improvements in AI models.*

People tell me like I talked to someone yesterday like should I still learn to code? Like is that gonna be a waste in five t years? And I'm like knowing how to design systems is going to be really really important and like they talk about like programming is building a theory like and building a theory and and and designing this stuff I think is really really important. One of the biggest improvements I made was just switching to Gemini Flash. You can see I I tried GD40 then Sonnet and then Gemini 25 Flash and you can see the difference it made just right there.

There's tons of data of cars driving perfectly fine on a nice sunny day on empty highways. That data is completely useless to every self-driving car company out in the world. What I want to see is a car carrying three other cars on a tow truck that looks like a car headed towards your direction with a median that's completely in the middle of the road cuz it's broken. That is useful data. And the goal with this is to say like, okay, how can we build a pipeline that takes all these different kinds of receipts and extracts the information from the receipts such as the item amounts, the grand totals, and everything like that.

Oh, wow. We're This is again, here we are again. AI that works. What's up, guys? How y'all doing? >> How's it going, Dexter? I'm doing great. I'm currently in an undisclosed location taking care of some business, but I

wasn't going to miss the pod because I'm very excited about the topic today. >> Where the heck are you? >> I am in an undisclosed location. I'm in a very colorful conference room. >> Looks like you're in a It looks like you're in a Willy Wonka factory if I'm completely honest.

>> You know, you're not far off. >> Well, guys, good to see you again. I think today's episode is one that I think funnily enough I had a few DMs this week just talking purely about multimmodal evals and I was like I was like going straight forward I was like oh my god this is a perfect episode for the timing of what's going on and then Kevin here who's who many of you might have seen from a previous Eval episode that we did had actually gone through and gone really deep into this problem and I was like well there's no one else better to have than Kevin right on this timing.

>> Well, I appreciate that and yeah, introduction for me like I yeah I was I was on the podcast gosh month ago, month and a half hard to remember lose track of time but yeah on a previous large scale classification pipeline evals but Kevin Gregory I work an ML engineer at Evolution IQ and we build claims guidance software for insurance companies. Hopefully we build AI that works. >> Yeah. I mean when you're in I mean I and that's a that's a thing I think that is like we don't we don't talk about enough on the show is like you know Vivoth has spends a lot of time and we try to bring on guests who are working in industries. It's a lot of like things that you can apply in like vertical AI. one of the things that led to like the 12 the whole 12 factor agents thing and the context engineering thing at the start was like, hey, like let's go talk to a bunch of people who are actually like shipping real AI products to the enterprise with high reliability in situations where like, oh, it doesn't work. Like let's blame the AI is like not an acceptable excuse.

Like it has to work. It has to work almost as good as deterministic software. >> I mean, if it doesn't, it's just not interesting. >> Exactly. That what what's the hard what's the hard problem, right? What's the thing that a lot of people may want to just like nope out on and how are people who need to solve real problems for serious businesses actually like putting pen to paper and and solving this stuff? >> And multimodal email is something that we do a lot at at Evolution IQ because there's a lot of medical documents that come in with insurance claims and you can OCR it and get text and just kind of treat it as a text input or you can just do multimodal. But how do you do that?

How do you build it reliably? Which do you choose? There's all sorts of considerations that go into making those decisions and and building out those pipelines. >> So with that in mind, I think what we should do first is let's just lay out the problem that we're working on for everyone so that way we can have it understood. So I'll screen share, I'll show off the excaladraw. And Kevin, why don't you just take us through and start posting some like general diagram of like the problem that we're investigating here together?

>> Sure. so you want me to start drawing an Excel draw? Yeah, >> if you want to talk through it, I can I can I can try to take notes if you want to do like the broad strokes and I I can annotate it. >> Yeah. So, many people on this call may be familiar with something called the chord data set, but what it is is it's receipt data. And the goal with this is to say like, okay, how can we build a pipeline that takes all these different kinds of receipts and extracts the information from the receipts such as the item amounts, the grand totals, and everything like that

and does so in a reliable fashion. And what's interesting about receipt data is that there's a lot of for one the the actual size of the data or the size of the images are kind of all over the place, right? Like receipts are, you know, everyone I think here's probably been to CVS and gotten the receipt that's like 30 ft long and it's kind of comical.

and and things like that are not at all what the LLMs are expecting, right? They're expecting kind of a certain specific size and dimensionality and so we're seeing >> you want to post some of the images here just so we know what we're looking at. >> Absolutely. Look so these were actually interestingly enough from Indonesia. So these are Indonesian receipts. >> Okay. >> Yeah. Oh wow. There's Yeah, look at that. So that's it took me a minute to figure out like why the commas and and decimals were different. and it's because it's Indonesian and you can see there's just so there's kind of a normal length one there and here is a really small one that with only one item and I'm just scrolling through and randomly picking them so I'm not kind of >> you know, so they're like not only are they receipt data, it's like receipt data that's like randomly blurred or like hidden away too. Mhm.

>> This is like redacted for privacy or what? >> I I suppose for privacy. Not a lot of vendor information here per se. It really focuses on the the totals themselves. this is just the data set. This is from hugging. It's a hugging face data set. >> Got it. >> yeah, I mean I can just >> I mean I can see how this is not only is this silly, it's like comically silly in the form of like CVS is like in that scenario you can't even see you can barely see the total. really squint >> and you can make out some pixels of what it might be.

>> I I put I put this one in. I don't know if this is actually this probably is not actually part of the data set cuz you cannot see the actual totals. >> I mean, if you squint that anyway, but I think the point here is like and some of them are like grease stains. Some of them are clearly have shadows and all sorts of other problems on them. >> There are some that are crinkled at different angles.

>> Yep. >> So, >> so like really real world re really really real world data is what I'm seeing. Mhm. Yeah. And it's another thing that's interesting is is and we'll kind of get into this when we start exploring and kind of discovering the things I did and the mistakes I made and what I found is some it seems like some of the restaurants randomly have different taxes that they apply and those can appear in different ways and don't always get added to the total. It looks like, see this PB1? You see this in this purplish one right here that I'm kind of moving around. Yeah, this PB1 is a restaurant tax that is only there sometimes.

and so yeah, >> you can add it to the total. You said >> sometimes it is, sometimes it's not. You can tell here it is because it's the only one that ends in a two and the total ends in a two. >> But it seems like sometimes it's there, sometimes it's not there. I also discovered that sometimes there are just discounts applied. so it's it's a kind of thing where the more you look at these, the more challenges you find. And that's kind of the point, right? Is you have to just start building the system and build a system in such a way where it allows you to easily and quickly uncover these things.

>> Okay. So, what is what does your output data set look like? Like I'm wondering like do you like have a table

model? Are some of these fields optional? Like what do you actually want in your structured outputs here? >> Sure. >> You said item amounts, grand totals, like do you have either a document or a a BAML or or something that kind of just demonstrates all of the things we might want to pull out of one of these?

>> Yeah, I've got a BAML file and I can just post quickly >> screenshots in here for now. We'll get to the code. >> We'll go dig into it later. >> Yeah. racted JSONs. If you have like extracted JSONs, they might be interesting as well to just take a look at really quickly. >> >> just so we can understand what the final end output is. >> So, this is the BAML class and this is of the final, right? I initially didn't have the unit discount or the rounding or things like that in there. You'll kind of see me discover these things.

>> Rounding. Interesting. Okay. >> Yeah. Interesting. Right. >> Yeah. I think this is just looking at this like my first gut instinct is just like like my first gut instinct is like I'm surprised that you need quantity for things like receipt data like this. I can see why but it's it's not how I buy most things. There's I mean sometimes they have quantities but usually I just say like what it is. >> Unit discount is interesting that you needed that in there.

>> Like this thing obviously flags me in a very weird way. the fact that you need this is really interesting. I I I really wonder why you called this grand total instead of total, but I I can see why you have subtotal. It sounds like >> that's it. >> Subtotal. >> Yeah, >> go ahead. >> Subtotal versus grand total. I wanted the LLM to be really clear on what the you know that there is those are two distinct fields and don't get them confused.

>> I see. And like we can look at this and we can clearly see that it's and it seems to be working mostly correctly. >> Mhm. Yeah. It's it's good. And there's there are some edge cases where I mean that you'll see that that when I look at the receipt I can't even figure out like what is going on this receipt. The numbers don't seem to add up. you know so it's it's it's very interesting. It's very interesting.

So before we go into this and really ask really ask okay so someone asked a question dumb question why did rounding stand out immediately well the reason rounding stood out to me immediately is like when I think of receipts I don't think of rounding my totals I usually just swipe my credit card and it the number is what it is so I don't I at least living in America we generally don't round stuff you might round stuff for tip and tax >> gas stations gas stations have fractions of a penny.

>> Yeah. Gas station, I guess. Okay. But that's rare. >> Or they used to. Maybe they don't anymore actually. Maybe that's like maybe I'm aging myself. >> And then so it just stood out as something weird that I would pull out because I it's just not a my gut instinct doesn't say that I would round by default, >> right? >> and then another question someone asked is why not do OCR and pass to an LLM? I think for that we have a really maybe we should just do OCR really quickly on some of these images and just to show what OCR does and the at least Kevin I'm not sure about your take on this or Dexter but my problem with OCR that I have always seen is OCR loses structural semblance >> yes >> whenever I do that. So like in the case of this thing up here in the first image on the top right over here if I were to do OCR I would get a one and LM dumpling chili SC and 68 0000 >> without the spacing.

>> Yeah I don't know I I would have to infer the spacing. I'd have to be like oh they're rotationally in the same angle so therefore it's correct. But if the image was taken at like a slight angle like this, all of a sudden I can't even use OCR to be like I have to go find like the normals of the image. And that's just a more complicated problem in my experience. >> Yeah. Okay. So I think I think probably for the rest of this episode like before we get to the code I think it would be really interesting to one maybe vib briefly recap just like the four or five categories of evals we talked about in the last eval episode of like runtime guard rails vibe eval like deterministic evals this kind of stuff and then talk about Kevin just really high level the architecture of your pipeline and then we can get into like what checks did you put it what parts and and how is it implemented? How's that sound?

>> Dex, I love that you're asking me to do this, but I pro I Kevin showed me a screenshot of his dashboard. I think you should just pull that up. It's going to answer that question really quickly. Let's just start dashboard that we ended up with. Kevin, >> the final one. >> The final one. And I think we can start with what we ended up with and then we can walk up to the journey of how we got there.

>> Sure. And like what was the process of discovery? Because I think I think there was something that stuck out to me that when you DM me is like I think one of the things that Kevin told me about this is like this problem was way easier than I thought. >> It was a lot easier than I expected. Yeah. >> and for people that were asking about handwritten documents or anything else along that lines like this problem is way easier than you think. But I think the key takeaway here that we had when Kevin and I were talking about this was it was only easy because the mechanism that Kevin used to break down the problem is what made it easy.

>> Okay. So the design of the design of the system mapped nicely onto the design of the evals because we had all that in mind from the start. >> Exactly. >> Yeah. And I I took a very similar approach to this that I took to the large scale classification pipeline, right? of what information is going to inform how you change the pipeline, right? Like what information is going to tell you where the errors are, what they are, and show you exactly, you know, what's going on. And then how do you display that in a way that's just that just knocks you over the head with how obvious it is what's going on, right?

So, this is the final one. Vivv, I ended up doing, you know, 350 receipts total instead of the 100 I showed you yesterday just to kind of fill it out a little bit more. and you can see here right this is the these are the the eval data completeness are there receipts and grand total grand total calculation does the subtotal I mean so these two grand total calculation subtotal consistency and sum validation are just looking at different pieces of if you add up just the transactions does it equal the subtotal does if the extracted subtotal plus the taxes and rounding does that equal the total so just basic summations that are supposed to happen. unit price accuracy, right? That is number of items purchased times the price should equal the the amount extracted for that line item. And then positive values, right? If you're extracting something, it should be a positive value, right? You're paying for something, it should be positive.

>> Funny that they're negative failures there. That's actually what's very surprising to me. >> Mhm. Yeah. And so I mean what we can do real quick is we can just look like okay so there are what two that fail the positive values so it's extracting negative values somewhere and that might be correct right the eval itself might be wrong

but we can just look at that so let's let's see if we go to the detailed results we can quickly just scroll to let's see this one a few failed that's not the so we can quickly just look to see where it failed with the positive. Here we go.

This one had positive values. or I'm sorry, this one failed the positive values. So, we just look at the receipt and so let's see. Are there negative values here? No, there aren't. Right. I'm not seeing any. oh, the discount. It extracted a negative value for the discount. And it extracted that as a line item, not as a discount. Because if we go here because we can show see the extracted data right next to it.

Yeah. It thinks it we purchased a disc and that it's not a discount on the amount but we purchased we purchased something called a discount. >> And what's funny here is that does lead you to having the right answer in the end. >> It does. Right. Because >> one and minus one on the row. >> That's right. >> Yeah. >> So because the summation all works but it's interesting. Yeah. >> And what's really interesting here is if you had for example, let's say you had built your software, can you scroll up a little bit where you did the minus DSC?

>> the minusc >> in the data set in the data in the extracted data. >> Oh. Oh, in here. >> Like what's funny here is like you could imagine someone saying, "Hey, unit price we know always has to be positive." And writing an absolute value on there >> and like programmatically and that would clearly lead to the wrong output here. >> Mhm. okay. So, if you worked around that it had negatives by just flipping every negative to positive and assuming it was an LLM error, you would actually break the thing because these two errors happen to cancel each other out probably like correctly structurally like from whatever system this came from. But yeah, you make assumptions that nothing is ever negative and you end up with Yeah. Okay. Mhm.

>> And what's interesting here is like this is just like one of the grant one of the failures here in terms of negative, but I suspect you're saying this Kevin because I suspect you spent a lot of time looking through the data and you every time it said gave you something negative, you're like, "Oh that's real world data, it's actually negative." >> Yeah. Yeah. Exactly. >> so question in the chat that I think is relevant. So none of these receipts have a golden data set, right? The hugging face data set doesn't actually have the right answers with it.

So the hugging face data set has it does have what they call metadata. I looked at it some and compared it. It was honestly it just would have taken a lot longer to incorporate into the pipeline because it has a lot of quirks to it that I needed to spend a lot of time figuring out. And I think my goal with this was to try to build a you know like in the real world right we we don't have the golden data set so how can we try to get closer to building that on their own was kind of the tack that I took with this but yeah hugging face does have what they call metadata which has a lot of information including the actual amounts.

My my gut says that for most people working on AI pipelines, especially like multimmodal data, they don't have a golden data set like what exactly what Kevin is saying. And I think if you go back to the original dashboard, Kevin, the homepage instead of the detail view, my first gut says it's really important for people to be able to almost elevate from like having no golden data set and only random data to first building a proxy of like is the system mostly working and which eval failure. So in this case like we looked at positive values even though positive values failing, it's actually not a true failure. It's a failure where if you look at it, it's actually correct. So,

we almost are like, "Okay, cool.

Positive values are a thing we'll spot check, but they're almost always going to be correct." Now, we can go look at some validation or subtotal consistency or grand total consistency. And what's interesting to me is even if some validation and subtotal consistency is wrong, grand total calculation seems to be way more correct. And being able to design from this and then slowly escalate to making a golden data set from this data is way more interesting than actually saying let's go make a golden data set from day one cuz it's just so much slower.

How by the way how long do you have >> take you timing wise? >> This whole thing probably took me three or four hours >> including running everything by coding the whole UI and everything. >> Yeah. It was it was really fast. Maybe I'm exaggerating, but it was not it was it was not a substantial time investment. >> Interesting. that's actually way less than I expected to be completely honest.

>> Yeah, this that's what I was saying when I meant that this is yeah, I want to say the stopwatch. I mean this is what we say about like code in in general is like I think someone was someone was posting that like oh code is now really cheap and software is really cheap and like update your priors about how and when and why you build software and one of my favorite comments was like the writing of the code was never the hard part like it's important to get it right but like when you have the design and and I know you demoed a similar dashboard to this so like you kind of already knew what you wanted and you knew how the system would be designed and you knew what kind of data you needed like formatted on disk and you knew how you would run it and it's like that's the hard part that I think takes a lot of iteration and time and like designing systems is still people tell me like I talked to someone yesterday like should I still learn to code like is that going to be a waste in 5 10 years and I'm like knowing how to design systems is going to be really really important and like they talk about like programming is building a theory >> like and building a theory and and and designing this stuff I think is really really important. Yeah, that's that's that's my take on like yeah, this was fast because you knew exactly what you wanted and you knew what the design would be.

>> That's a good Yeah, that's a good point. >> And that's that stuff was hard earned. That stuff probably took months or years to develop. >> Let's let's go back to day one. like when you first started this project, Kevin, what was the first thing you did and what did you end up doing next? >> Like how did you end up in this final design in the very >> Sure. to the very first >> I I'd love to know. Yeah. Okay. Sorry.

Tell tell the story. I'd love to also know like a little bit more detail like how it actually works like not every line of code but like how do the different components of the system fit together and like what are the interfaces that you created to make this make this work well for you and be kind of like be able to evolve it. But >> sure. >> All right. Let's go let's go to baseline number one. 21 receipts. Okay.

>> Yeah. So I started with just very basic like training wheels, right? like I don't want to spend a lot of money on LLM compute if nothing's working. So this is using GPT40. and right out of the gate and you can see that the amounts aren't it's okay but there's a lot of mistakes, right? So the sum validation is the biggest one that we're missing. And if we we look at that we just can look at one of these.

It's so interesting to me because it's so it's so tempting to think that an to forget that LLM are just math and computers behind the scenes and there's not they're not actually people because you'll just see flat hallucinations here that are just plainly wrong. I mean I don't know one right off the top but it's missing something here. You can tell it's off by a lot right 173 200. and if you would kind of scroll down the extraction here and the the receipt, you'd find that there's just one that is just completely missing or just completely wrong. So my first thought is, okay, so what if I just use a smarter LLM, right?

So instead of using GBT40, what if I use >> before you show the results of a smaller smarter LLM? Question. Did you have all these eval designed from minute one? Yeah, I did. My So, my thought was so if I'm extracting a receipt and I'm getting things like the subtotals, I'm getting the the item amounts, the grand totals, I actually went back and forth with it was it was sonnet and cursor and said, "Here's kind of what I'm doing.

Like, let's brainstorm, figure out what some good runtime eels would be." >> Okay. So, the first thing you actually did wasn't actually do this. You just stopped and thought about the problem for a little bit. So the first thing I did was look at the receipts. That's the very very first thing I did. I downloaded the data. >> Look at your data. >> Looked at the receipts. That was Yeah. And that's when I realized that, hey, this is this is not this is not American currency, right? We're we're somewhere else.

>> Okay. >> so yes, the very first thing I did was looked at my data. Just spent some time looking just like we did with the the the whiteboard, right? Just looking at different receipts and wow, there's all these kind of different things. Some are grease, some have some handwriting, some have random discounts. Well, I mean, I didn't see that right off the bat, but >> okay, >> looked at the data. >> What what I what I love about the design of this so much is you didn't have to do any hand labeling. You needed no golden data set. You designed a system to evaluate the accuracy of extraction solely based on like the invariance that you know should be true about the receipt.

>> Exactly. >> Okay. So, what you So, you looked at the data. You literally I'm just saying you just download data and just scroll through images and just like pick random ones and just like skimmed really fast. >> That's it. >> Okay. So, step one looked at data. >> Mhm. >> Step two, what did you do next? >> Step two, I set up like basically set up the the project, set up the repo, set up BAML, and went back and forth with an LLM to figure out what runtime evals there should be. So really quickly, you what do you mean by set up the project? So you does that mean you started loading the image files, you started being running a small test harness in Python where you could like loop through images really quickly or was it purely just like initialize like >> it was purely just initialize.

>> Okay. So not really anything just so you could have a folder to work out of. >> Exactly. Just got so I got a folder to work out of. >> Okay. And then I'm guessing you defined your receipt data model very curs like very trivially. >> Yeah. define the receipt data model in in BAML. >> >> okay. So the original one didn't have all of these like rounding >> what the original receipt data model was. Do you have that somewhere?

>> No, I don't have it. >> You can just write it if you just want to write it really quickly like to be like receipt v1.

I'm just really curious what you ended up. >> I mean, I can just kind of pretend here, right? So, this is what it ended up being. So, the initial one was literally just item name, quantity, unit price, total price, and then for the that was the transaction data. And then for the receipt data, all of this was gone. And I just had transactions subtotal.

no, I think I just had transactions in total initially. It was just add up all the transactions that should equal the total. >> Got it. Okay. And then then you went through like a cursor conversation from here and you said what are some runtime emails >> I can do? >> And then that got me to update this. So I had the subtotal and the tax which made sense to me. >> Got it. And that was like it didn't it didn't really like disagree with what you were thinking. It was like this seems obvious and the runtime the cursor conversation led you to have and if you pull up again what what eval you were showing.

>> Mhm. >> what the eval you had were data completeness grand total calculation unit price accuracy subtotal consistency positive values and some validation >> right? and then that once it described those you added subtotal and tax and now you have a now you have a data model and then evals that you have. >> Exactly. >> Perfect. Cool. And then you ran that on a very cheap model. I guess the model that you're most familiar with which is GD40. I like it's not even about cheap.

It's just about familiarity. It's just like the model that you probably it's your go-to model for a task. >> Can can we pseudo code out kind of like the core loop here? It's like for each image. I mean, I I guess it's pretty clear, right? You take each image, >> you you you you you run the extraction, you do the math, and then you record which of the which of the checks passed and failed. >> And they're all just pass fail.

>> Exactly. They're all just pass fail. Yeah. >> >> Okay. Do you want to show that? Actually, that's a good idea. You want just want to show that? >> Yeah. It would be interesting to see the code that works. It's going to be in the AI that works repo. But do you want to show the code really fast? >> Sure. or like show yeah show one of the evals or one of the like just like the the code that like takes the output and does the math on it. I mean it's it's pretty simple code I'm sure but it' be kind of interesting to see it for real.

>> So let's see it's all it's all zoomed in so it's a little little off. so >> you have an image you produce extracted data on it right there. >> Right. So this this is the extracted data. and you have error handling to be like sometimes it fails, which is also fail. Which is also fair. Yes. >> Yeah. Which is actually the the dashboard keeps track of how many failures there are, >> which spoiler alert, I tried to do Gemini 3 last night and I got a ton of extraction failures. So, >> Got it.

>> Yeah. Then not sure what's going on with that, but something to figure out. >> They said this one's supposed to be better at tool calling. >> I don't know. Maybe maybe it is. maybe I was doing something wrong. It's very possible. >> No. I mean I'm I'm sure they said that and it's not as true as they want it to be. >> Yeah. >> Okay. And then you produce an evaluation result. >> right. And if we just look at say evaluate grand total calculation.

>> Cool. It's just like >> you're just doing math on a JSON object. >> Yeah. Okay. Cool. So there's like there's no

there's nothing fancy here. You're literally just doing >> nothing fancy. Oh, tolerance is interesting because you have floating point numbers. Makes sense. So, you had to go real tolerance out. >> Mhm. >> Did you have tolerance from day one or was that something you added later when you saw some of them were like off by 1 cent?

>> I had this from day one. >> Yeah, >> I sus. Yeah, if you're ever doing floatingoint math calculations, you will always have this error. You need like you need a tolerance. You don't have a choice. >> Cool. yeah, it's very it's very basic. Like I said, this task was it surprised me as to how easy this task ended up being. I was expecting a lot more kind of I was going to have to spend a lot more time on it.

>> And you know what I find really interesting about this is if you wanted to add another email, it's actually really easy for you to add one here >> because like you just add one more to the list. It's effectively zero cost. >> That's it. >> That's cool. >> Yeah, that's it. So I can see why you said this basically took you three hours because you basically have two separate pipelines here. You have one pipeline that does the actual extraction. You have a separate pipeline that runs the eval on those platform on that extraction. They're both disjoint. They have no dependencies except the shared data model between them which is the receipt data object that you showed us in the receipt. AML file. And then you have a third system that visualizes the results of the second system.

>> That's right. And you just have a data contract between them that shows how to go render everything. >> Mhm. >> And last, >> okay, so the evaluation results get written to like a JSON file right next to the extraction results. >> Exactly. Yeah. I mean, if you look results, we can look at this one detailed results. This is if we scroll up, you see the evals. This is what the the streamlet app is reading from here.

Okay. So this is for a given receipt for an image path. So this is how it lets you render all that stuff if you need to. >> Exactly. >> And then Okay. >> Exactly. Cool. >> And that's how he loads data dynamically. That's how he pulls up all the information about it. >> It's >> And is the extracted is the extracted data embedded in here as well >> in like this JSON object or does it have to look?

>> That was that was what I pasted in the white. Yeah, it's right down here. So yeah, it can just read this and the stream lit app has everything that it needs. And the reason that this was so fast for you to do Kevin from what I understand is last time when you built your classification system you actually spent a lot of time on designing this shape >> like you're like what is the shape of this data extracted data has here there's a bunch of evals that have these names and these results and then it has to have the model information because I want to be able to compare same image on different models it has to have a run information because >> I might run the same thing multiple times based on things that I changed along the Okay, so you're basically just shaping the data shape for the tooling before you actually really build it. But once you've designed the tooling, it's effectively zero work to make any different system use the same tooling.

>> Yeah, you know, that's actually a really good point. I hadn't realized until you just said all that how much my work on the previous one kind of set me up for this to go really really quickly. Yeah, that's actually very similar to how I have seen most AI like most companies that we've worked with have actually had a very similar

response where like I think the work up front feels so painful and so annoying cuz you're like why am I doing this? I can just like one hack this like not think about this and just do it oneoff.

But it turns out if you do one-off work every single project takes the same amount of time consistently. But if you do do the upfront work up front where you just stop and think about the design system a little bit better, the next project similarly just takes way less time because most of the fundamentals are truly the same. Now I'm curious from the design. Oh, go ahead, Dex. >> And I actually just to like echo your point. I I really like this pattern. I naturally stumbled into something like this when I was building like a PII extraction and like scrubbing pipeline where like I was writing after each step of the pipeline you want to write the JSON because then you the human can inspect it. You can resume from a past result. You can test incremental parts of the pipeline like the results actually can become like the bits that you use to build more like baked golden evals, golden data sets, golden like test sets so that you can you can know that and like having JSON is nice because it's human readable and machine readable for some some pe some people J say JSON was meant for humans. I don't know if I would go that far. JSON was meant for was made for machines.

>> JSON was meant for humans. If machines are the only thing we cared about, we'd all use protobuff. >> That's all right. Fair enough. Yeah. anyways, no, I I think the structure makes a ton of sense. And like I can't imagine building any kind of AI pipeline. My question actually for both of you is like do you have thoughts about how this would scale? cuz like once you have a 100,000 images, is it actually like performant to do this in JSON or do you have thoughts about like would you would you move this to a like obviously same structure and like checkpoints along the way but like what are the limitations of doing it this way?

>> Well, I don't think JSON itself is necessarily bad. You could store into an S3 bucket instead of JSON. It's the same thing like like it's S3 bucket with pads. The fact that you're using a file system is the storage layer itself doesn't matter. What if your results gets too big to like store into memory? Like you have to then figure out how to like you have to do some kind of like sharding. Yeah, but you need to pull it down to do each incremental step of the pipeline >> that oh sure put into MongoDB database then like put into MongoDB database and like query only the fields that you have like Kevin did over here. If you scroll up Kevin like the the JSON strct that he's storing is basically scroll up it has a thing called evaluations.

Literally you can pull everything but the extracted data and only pull the evaluation side of it which should be small enough but and we all know how to do like >> but I mean if you have 500 million records you I mean I that's probably too high to be reasonable like that's the number of like >> even with that like we know how to do pageionation on databases like >> you can't do that in S3 though like I I agree like I you need that is my kind of my question is like you need something that supports like slicing and filtering and pageionation right >> S3 does that too like AWS has built a bunch of software on top of S3 that has all the querying pagidation as I'm not saying you should use S3 necessarily.

It's just a different like you can solve this problem as another engineering problem rather than having to think about like saying I have a bunch of data that is somewhat structured and I want to query it with some aggregation is a well-defined problem that I'm certain cloud code could solve. >> Okay, so VBO thinks my

question was boring. >> Well, maybe >> I can tell you though if I had 500 million records I would not I would not be using a stream app. No way.

>> Yeah. Yeah. No, this is I mean like this feels like a really good Have either of you ever worked with parquet? It's basically like gent json. >> This would be great for like or like lance. >> I'm sure people are already is there. >> Say what? >> Or lance DB or something. This would be great for that. >> I don't know if enough about lance DB to comment. is really good for like multimodal data sets on top of it which makes it really cuz it does like the oneoff links to not saving in your actual data.

>> Now I have one more question Kevin what did change in this pipeline versus your previous pipeline that you made? Were there any architectural changes you did have to make? the I think the the biggest one was in the previous pipeline we had multiple checkpoints because that we had I mean I don't know how many people on the call were were a part of that but it was a large scale classification pipeline where the first thing we did was we dumped a bunch of categories into an embedder to filter that down and then we took just the top however many categories and then dumped those into an LLM with the actual query and then we get the final response.

So, we were able to check each one of those steps, kind of what's going in and out of each step and figure out where the problem is. Here, it's it's kind of just a one shot, right? There's no break points or probes in order to to check and see where things are kind of breaking down. it was one prompt. I guess you you could kind of say with the different evals, there are all these kind of different little points, but still there's not the it's not the same.

Oh, I have multiple checkpoints here. I think that was probably the biggest one. >> Got it. Okay. So, the fact it was like a structurally a different problem because you only had one checkpoint and no incremental progress along the way to measure. >> So, you weren't analyzing multi-steps, you were analyzing one. So, I'm guessing your JSON shape did change to represent that. >> Definitely. Yeah. >> Okay. >> And and the last one, didn't you also have to handle label the data? Like there wasn't like an answer key for this stuff, right?

>> Yeah, I had to handle label the data. And last one there was no real way to do at least that I could think of runtime evouts. >> Cool. >> I remember reaching out to you know my family members and me and handle label the data and say it was it was items in a a hardware store and what basically categories they should fall into in the hardware store. Another thing that's interesting about >> another thing that's interesting about the previous problem is that the previous problem had multiple right answers. That was something that we found that was really interesting in that previous one was, you know, like I I don't remember any of the examples, but something such as I don't know, I'm blanking, but like in in an air conditioning filter could be an HVAC or it could be in >> in air conditioning.

>> In air conditioning. Exactly. And those could be two two different categories. And so what was interesting last time is we went through the mistakes and we actually we we said, "Hey, these actually kind of are correct." So instead of having one answer, you have a set of right answers. And we would check to see if our output was in that set. Here there is a right answer, right? Like they paid a certain amount for whatever you know whatever they ate. So that was a different kind of way to think about it as well.

>> That's actually really interesting. >> what did you I was going to say like so what did you use this? We're getting a little short on time and there's one good question in the chat, but like what did you use this for? Like did you actually take the eval and then go back and try to improve the models and switching the models? Did you change up your prompt at all? did you were you able to use this to drive improvements in the extraction? Yeah, let's look at the prompts.

>> Yeah, I can show you. Well, I think yeah, so here's the actual here's the prompt that that's not it. You can see I I played around with extracting number of transactions, but I didn't need >> as a pre as a pre-step. >> Exactly. didn't end up needing it because this worked so well. And this is the prompt, right? So each transaction or each item, this is what you want for each item on the receipt and then all these receipt totals, right? And and these didn't all I didn't discover all these right out of the gate, right? Like we said before, rounding discount on total yeah those didn't like I didn't have those right out of the gate. Those came from kind of iterating and experimenting.

>> Okay. So you iterated the data structure and the prompt together because of this thing that like we do all the time on this show which is like prompting through your output format basically. >> Right. And I mean we can we can see here if I go to let's see it's I think it is yeah it's this one. So if we load this which side note one of the biggest improvements I made was just switching to Gemini Flash. You can see I I tried GD40 then Sonnet and then Gemini 25 Flash and you can see the difference it made just right there. going from 40 to Gemini Flash almost made this only has one mistake. So if we look at the mistake, you can see it's here.

Surprise surprise, it has a discount of 19,000 and I mean now the discount, you know, the discount's here because it's part of the data model now, but before there was no discount and so it was missing that discount amount. >> Got it. And you can see the difference is the 19,000 which is the discount. So it's like so that's when I saw I said >> you started with a small set of receipts. You figured out what can we learn about making the data model and the prompt better with that small set.

And then once you got those pretty pretty good and you said even if one of these is failing right you can basically say like okay that one we're not going to try to solve. Let's do a bigger data set. Let's see what other problems we hit. And so you built a tool that basically like when things are not passing, you can immediately dig in and use what you learned from the eval to go improve the prompt.

>> Exactly. >> Go find more corner cases. I love it. >> You can you can kind of find like see my journey just by looking at the named runs, right? So we're here at Gemini Flash. I added just a discount on total field and then I noticed that there's some item discounts. It's like a percentage of the item. So then I added that and that looked pretty good. So, I jumped up to 50 receipts or 51 cuz I forgot it started at zero. Whatever.

>> Retry logic. >> Then I added retry cuz I was just getting extraction failures. I'm like, it. Like, let's just do exponential retry. And then that one. >> Can we see the receipts? Can we see the results from each of those >> like the 50 and then with the retry logic? >> Sure. >> Like I'd love to kind of just like see it progress over time >> just like the highle analysis. Yeah. >> Yeah. >> Okay. So here and then if I do the retry added, you can see the

unified accuracy.

Yeah, it just it just got even better. >> Oh, sure. Yeah, cuz it's just like Sure. There's just some weird flakiness. Let's just like run it. >> Yeah, exactly. >> Cool. Okay, go on. >> >> and then what was the next one? >> Next one was same thing but 100. >> And again, similar performance. It's doing well. And this is where you get to the point where and viab this is what I should this is the one we saw yesterday. and this is where we start looking at the mistakes. It's like I don't know how I would label this.

Right? These these are the interesting ones. All right. So if I come down here I mean we'll just look at at this one. difference of 3,000. Let's see if this is an interesting one. >> Really interesting. Like certainly Kevin hasn't looked at this data on the fly. He's literally looking at it right now and it's like I see something is off by 3,000. Mhm. >> And like here I can see that it literally double counted the 3,000 of the discount and the tax.

>> Wait, what's the discount? >> Yeah, >> it just added a $3,000 discount. I have no idea why. >> Oh, yeah. I see that. Yeah, >> I think that's Well, what's also interesting is like this. >> What are the 50,000 and the 17,000 underneath the cash that they >> paid? Okay. Okay. Okay. So here it double counted that. Yeah, it's >> So I would probably iterate on the prompt on this one, but if we just look at a couple others.

>> Yeah. Okay. >> Like let's see. Oh, this I think this is that discount one that I got confused on. >> Yep, there's that. >> >> I think what's really important here is I want everyone on the everyone on the call to really quickly realize how fast we're looking and understanding this data. The key part here is understanding the problem. And I think someone in the question, someone in the chat asked that important question is like, isn't this stupid? Aren't we doing manual prompting? Shouldn't we do like an optimizer? And in theory, you could use an optimizer, but the real problem is the reason that we can't use an optimizer cuz real world data is messy.

You can optimize if you know exactly what you're optimizing on, but we don't even know if it's correct. Like if our emails are actually correctly defined, like in the case of earlier in the chat, we talked about like negative values. We did see correct negative values applied >> and if we were optimizing on that failure the prompt would be like don't add negative values ever but that doesn't actually mean that that's true. So while an optimizer can be useful, it's only so and so that's useful once you understand >> like it will overfit for what you are telling it to optimize for. And if you don't have good definitions of the final outcome, you will lose >> like it will >> it would be cool to take this data set and run it through a prompt optimizer and see if it can improve the eval performance. That might be a fun We don't have time to do it today, but I thought I like doing like a Jeppa or like doing the like BAML DSPI like Frankenstein pipeline that someone's someone talked about.

>> Yeah, I think it's really important like fundamentally regardless of what you use, it's really important that people look at the data. like the tooling that you build around looking at the data. While it sounds stupid and silly and slow and arcane, this was actually the thing that'll help you speed this up because the real thing you want to optimize is a data set of 10,000 receipts. You don't want to optimize on a data set of 100 receipts.

And if you think about it, the best example that I think is very tangible for most people is actually self-driving.

So when you work in the self-driving space, there's tons of data of cars driving perfectly fine on a nice sunny day on empty highways. That data is completely useless to every self-driving car company out in the world. What I want to see is a car carrying three other cars on a tow truck that looks like a car headed towards your direction with a median that's completely in the middle of the road cuz it's broken. That is useful data. And it's the same thing in here. When I go and build like the prompt optimizer, what I really want to do is I want to find the data that is relevant to then go build the optimizer on.

Like that's what is the real fundamental question like how do you find like the most odd data sets that are actually going to help me decide this and then you can go ahead and build what I would say is like turn this into a golden data set and say hey I found these weird edge cases let me go and define the perfect JSON for each of these data sets. >> Mhm. >> This is exactly what the final output should be and now go eval that against that for these specific very small data sets that I have. And I I think to your point VA is this isn't we're doing this really quickly, right?

Once I once you've done something like this before, right? Building a building it again for some a different system is fast and now we're iterating through it very very quickly. Like this whole thing understanding the problem space a lot better. You can do in half a day or a day tops. and then you're much better equipped to do what you're saying and build your golden data set, build the right JSON, and then maybe do a prompt optimizer. But it doesn't take much time in order just to, you know, invest a little bit up front and then you'll have that to inform the decisions you make down the road.

>> Yeah, it's really about like I think it's really about like deep understanding of the problem. >> I don't know if you want to keep talking about the optimizer thing, but like there's a question in the chat is like a human won't be able to find the best prompt manually. And I think like I want to I want to double down on sorry, what did you say? >> I I mostly agree. >> Okay. It's it's like it's almost like it feels like it's like this this perfect world framing of it where you have access to infinite data, every single potential thing that you might hit ahead of time. then yes, like a prompt optimizer will do better. But I also think like by under optimizing you lean into the like emerging capabilities and generative nature of these systems where it's like you don't know exactly what it's going to be capable of and you're you're better off prompting less and less specifically and having a good feedback loop like we've built here.

You know what I would do here is like let's say I was shipping this in a product for actually making this like for like autoesting receipts on like Brex or a bank banking app or like any sort of like Pinops application like Concur or any sort of like receipt management system. What I would do is I would go if you look at the top level data set can you go up Kevin? >> >> yeah like let's say I'm filing like a reimbursement for my company analysis.

Oh, yeah. This one. >> What I would see here is like, look, what I want to ask myself is, I want to ship a product. I don't actually care about hitting 100%. Here's what I care about. I care about the user's problem being solved. The user's problem here is entering receipt data is freaking annoying and really hard. >> So, here's what I would do. I would look at this data and be like, "Okay, cool. We're hitting a really high percentage success rate."

Like, it's mostly correct. what's the exact percentage here? Do you know what it is, Kevin? If you scroll up over grand total. >> Yeah. If you can save me having to enter in a receipt >> 3% failure rate like I'm at a 3% failure rate 99% of the n over 95% of the time I won't have to enter the receipt. What I would say is great I will ship this app but because I have all these guarantees built into it. What I will do secondarily is into my UIUX I will build a system that says something else which is I will say I will flag that for the user and say I found a mistake. Can you please double check every single entry manually and I would literally force them to check check check every single thing in the UI to make sure they actually validated against the actual receipt. And now I have a system that's 100% correct.

>> And yeah, it's it's about bridging that gap with human in the loop, right? It's like as long if you're saving me if I only have to do that one in 20 receipts, you're still saving me a ton of time. Cuz without the system, I would have to do every single one of them. >> Not only that, you would have had to manually enter it versus checking for accuracy. Huge difference. >> And only checking the ones that fail my checks, >> which is also a huge different shift.

So, like the burden just went from like uploading receipts being like a painful task that takes like a couple minutes to being something that takes 90% of the time one or two seconds and then 10% of a time takes a 30 more seconds on top of that. So, my burden is way less. But I could go even further. What I could do, we could build a second system here that says the LM is actually going to be wrong. We'll assume that the model will be wrong. And then we'll build a second system on top of it that says whenever we get a grand total calculation error.

We'll actually tell the model, hey, your error is wrong. Your grand total is completely wrong. Here's how much it's off by. Update the original data model to produce here's the original data model. Here's the error that you have. Re-update to go do that. And now I can run the grand total calculation again off of that error. So, not only building in the runtime checks as a as like a thing I'm doing for evals, but actually building into the product and saying when >> as a just like self-correcting like hey retry because there's an issue with this kind of thing and not even sending it to the human >> and then if it and I let that run up to three times and if that fails the third time I send it to the human >> or I might even show the human the UI and let the human know, hey, I found an error. I'm working on fixing it. give me a give me like a second and I'll fix it >> and the human can basically then review either review or not fix it. It's up to them and that's kind of like a few other things that you can do here. And I think it's more about understand these emails are not purely about like offline emails but how you can make them be online so that you don't have to prompt optimize and end up with the perfect prompt because if you can only ship your product when it's perfect, you will lose the battle of shipping product. Yeah.

>> Yes. >> That's a great takeaway. >> yeah, >> Kevin, if you want if you if you had one piece of advice to someone who wanted to build a system like this, like what's the what's the one or two biggest takeaways from your side? >> Gemini, so the first one is Gemini Flash is seems to be the best at OCR. So like not notably better than Sonnet or 40. So just going from 40 right here, same prompt, same data model, everything to flash right away, noticeable.

>> Yeah. so that was the biggest thing that surprised me. And the second one, I mean, we've said it before, but it's it's you got to look at your data. you won't I mean maybe some to some people the rounding the discounts the different taxes maybe that would be obvious to some people but particularly the discounts and the rounding weren't obvious to me even even after I looked at some of the receipts initially I still missed it I didn't check 100 right and so it took building this out and then looking at the errors and seeing like okay I understand what the error this is making and you know this is obviously going to be present in a a lot of receipts because these receipts just tend to have, you know, this data tends to have this feature. So, it's it's looking at your data and there's no real magic way around that that I found.

You have to understand the problem. >> And what's really interesting about that is it's like changing the shape of your data isn't just like changing the prompt. It's actually about changing like the data model that your code is using around the system. >> Okay, question for you guys. knowing what you know now. now you don't have to name any names, but there's a lot of companies out there selling evals, either selling the problem of you must be doing evals or selling products that help you do eval so you can improve your stuff. What do you think about evals as a business?

>> I think obviously everyone wants to make money doing something. So there and there's not it's not like it's not valuable, but I think it's very similar to how front end works. You don't really buy front end. You can buy someone to build your front end. You can buy someone to host your front end, but you don't buy your UI components. Typically, your UI components are yours and your businesses. I think Eval is very similar. You got to design the eval like the metric itself. Anyone that's telling you is selling you a metric, it is scamming you because the metric is so domain specific, so problem specific that it it doesn't really matter. And then everything else is just like harnesses to run stuff.

That's what Joshy just said. >> Oh yeah. >> Aren't existing eval solutions mostly harnesses to run? I mean, I remember when Brian came on and he was talking about their decaying resolution memory and he was showing some of their code and we was like, "Hey, are you okay sharing kind of some of your closed source stuff?" He's like, "Yeah, I can show you guys the code. That's okay. I will never show you guys the evals. The evals are the thing we keep super tight." And it's like, "Okay, yeah, that's actually the hard work of building the product is like developing over time in the same way he didn't want to outsource his memory system. He didn't want to outsource his eval system because it was really really tailored to his product and and and his problems and his users.

>> Yeah. And I'm not saying there's not value in paying someone to run your evals for you. >> but I'm also not saying there's like a necessary need and an urgent need to go do that either. like in my opinion like what Kevin just did over here was like clearly didn't take him that long. It did take him some design time and some system design time. I guess if people use his source code and point it point cloud code at this repo and say hey design me an ebos system works kind of like this or like chat with chat with look at this code and help me think about how to design evals for my own system like what design system I can use there. I'm certain they could do it in maybe not three hours but probably not one week either. It's probably like a one day process to go design this out.

And like my my thought process is like just do that. And then if you decide that hey this is we're running evals on 500 million data sets and we need to run like an offloaded distributed system and we don't want to own that. Great. Go pay for that. If you're running like 500 receipts just run on your stupid machine. Like like it's like async.io is not going to break on your system. If you want to have a shared distributed system that everyone can see these results and you don't want to go build that for your team, then just go do that. It's not going to take you that long to go do that. But also pay someone for that. That's not a bad thing to have. Like like building out this versioning system, if someone has designed it in a way that is really beautiful and good like Verscell has done a great job at shipping front-end UIs with staging environments on pull requests, like all that stuff is really really good in Verscell. has nothing to do with writing front-end code, but it makes writing front-end code better.

>> Exactly. And you can build your own system for that, but like I don't want to I don't want to say like for a PR launch this preview URL. >> We built our own at Sprout. It was it was it was big. It was incredibly valuable though. Like it was it was the most useful part of the dev platform at the whole company. >> Yeah. But it's so much better to just pay someone for it. >> and I think that's kind of what it comes down to. It's like you got to pick the parts of your eval system that are actually useful. If you don't have like a hundred people looking at random evals results all the time, then you probably don't need this. I would just go ahead and straight just like host it and just send it over like send over like a tail what's a tail scale URL to your teammate and go do that. Produce a bunch of JSON files that you can share over some like check them into git if you want. It doesn't really matter.

and I think it's just about designing the system you want. And like paying for it, I think can be useful, but it it also isn't like a necessary thing that you have to do. Emails are necessary, paying for them or not. >> Amazing. this was super fun. Kevin, thank you so much for jumping on and sharing. I can't wait to >> seems like about every six weeks you've gone and change changed the rules of the game. So, hope hope to have you back again soon. This is great. any any last thoughts? All the code is already on GitHub, I guess.

>> I haven't I haven't pushed it yet, but I'll do that. >> Push it. Make the PR. We'll merge it in. I guess for everyone else that's listen still listening. this is AI that works. If you guys are interested in this kind of concepts and you like seeing this kind of content, come check out the subscription over here or check out the YouTube. We'll usually post the videos one week afterwards. Really appreciate this time with Dex and obviously Kevin for making up the time.

It's been always a wild ride and thank you everyone for joining the chat as well. >> Yeah, thanks. >> Thanks everybody. >> Bye everyone.