Conversation is the New Middleware

The thing I find strangest about the current wave of language models is not that they can write. It's that you can keep talking to them. Change the subject. Come back to the old one. Ask them to rewrite the thing they just wrote, but shorter, and in the voice of someone who hates you. They go along with it. Nothing we had before did that.

People keep describing these systems as search engines that write paragraphs, or as auto-complete on steroids. Neither description gets at what's actually new. What's new is the conversation itself. And conversations, it turns out, are software.

I didn't see this at first. When I started poking at LLMs, I thought I was using a fancy text generator. But after a while I noticed that the part I cared about wasn't the generation. It was the loop. You say something. It says something back. You correct it. It adjusts. You pile on constraints. It holds them. Somewhere in the middle of that loop you stop treating the model as a service and start treating it as a place where work gets done.

That's a strange shift, and worth thinking about.

Most of the programming tools we've built in the last forty years assume you know what you want before you start. You define a schema. You write a function signature. You pick a data format. Only after all that do you get to express the idea itself. If halfway through you realize your schema was wrong, you stop and fix it, and the cost of fixing it usually grows with how much code you've already written on top.

Conversations don't work that way. In a conversation you can start with something vague and make it sharper as you go. You don't have to know what "the thing" is before you begin describing it. You can mix registers. You can switch between "tell me about X" and "now give me X as JSON" without leaving the session. What would have been three tools and two meetings is now one exchange.

For programmers this is unfamiliar. We're used to structure being expensive. Here, structure is something you can ask for at the end.

There's a second thing that happens in conversation that I think is underrated. Language models translate between ways of saying things. Not just between English and French, but between "customer," "user," "account holder," and "the person on the other end of the phone." In a normal system, if two services use different names for the same idea, someone has to write a mapping. In a conversation, the mapping is implicit. The model will treat them as the same thing until you tell it not to.

This sounds like a small convenience. It isn't. A huge amount of what we call "integration work" is just negotiating vocabulary between systems that were built by different people at different times. If you can get that work done by asking, a lot of middleware becomes unnecessary.

Which brings me to the word "middleware." It used to mean the layer of software that sat between applications and made them talk to each other. Message buses, ORMs, ESBs, the whole bestiary. The point of middleware was to standardize. To turn a mess of incompatible things into a smaller set of compatible ones.

If you look at what a language model actually does in an application, it's doing that. It takes whatever you give it — a user's half-formed request, a log line, a pile of JSON, a question in Portuguese — and produces something another part of the system can consume. It glues things together. That's middleware. It just doesn't look like middleware because it's made of words.

Once you see this, the shape of outcomes starts to make sense. Most of them give you three things. You get knobs for controlling how the model behaves — how chatty, how cautious, what to avoid. You get a way to run a chat loop and optionally remember what was said. And you get hooks for feeding the output back into the rest of your system.

Knobs, a loop, and hooks. That's not a content generator. That's a component.

The frameworks springing up around these LLMs — LangChain is the one people mention most — are basically trying to turn the component into a platform. They handle the prompts, the memory, the chaining, the glue between multiple models. They're doing for conversation what application servers did for HTTP: wrapping a raw primitive in enough infrastructure to build on.

I don't know which of these frameworks will win, and I don't think it matters much. What matters is that people have decided the conversation is the thing worth building around. A few years ago that would have been a strange idea. Now it's the default.

There's one thing I think will matter more than the frameworks, though, and it's the piece that's currently missing. Conversations are great at being flexible, but flexibility isn't always what you want. Sometimes you want the system to know that a "customer" is a specific row in a specific table, not whatever the model feels like meaning today. You want it pinned down.

The obvious place to pin it down is a knowledge graph, or something like one. A structured model of what things are and how they relate. Combine that with a conversational interface

and you get something interesting: a system that's loose at the edges and firm at the core. You can talk to it in any shape you like, and it still knows what a customer is.

I suspect a lot of the interesting work in the next few years will happen at that seam. Not inside the models, and not inside the graphs, but at the place where fuzzy language meets hard structure.

If that's right, then the mental model people should have for these systems isn't "an AI" or "a chatbot" or "a writing tool." It's a piece of middleware, sitting between your users and your data, turning one into the other in both directions. The novel part is that it does this by talking.

Which raises a small, funny thought. For decades we've been trying to teach computers to behave more like rigorous logical systems. And now the thing that might finally tie our software together is the opposite: a component whose defining feature is that it can hold a conversation.

Maybe that was always going to be the answer. Most of the work humans do with each other gets done that way too.