Unbundling AI ChatGPT and LLMs can do anything (or look like they can), so what can you do with them? How do you know? Do we move to chat bots as a magical general-purpose interface, or do we unbundle them back into single-purpose software?

5 October 2023

ChatGPT 3.5 launched a year ago, and I think we're all still working out what the questions are. The AGI argument happens in one corner (now mostly ignored by everyone else) and the semiconductor, model size and running cost conversations in others, but I'm most interested in product. How is this useful? What do you do with a technology that promises it can do anything?

This summer, Bill Gates said that when he saw the first GUIs at Xerox PARC in the late 1970s, he realised that this could be a step change in who could use computers. You wouldn't need to learn keyboard commands anymore - instead, you could just click on what you needed. However, you still need someone else to have created each individual tool that you're clicking or tapping on: someone has to make the buttons. LLMs look like another step change in the level of generalisation: one piece of software could become any tool.

We had a false start towards this with the wave of voice assistants that launched almost a decade ago. Alexa and its imitators mostly failed to become much more than voice-activated speakers, clocks and lightswitches, and the obvious reason they failed was that they only had half of the problem. The new machine learning meant that speech recognition and natural language processing were good enough to build a completely generalised and open input, but though you could *ask* anything, they could only actually answer 10 or 20 or 50 things, and each of those had to be built one by one, by hand, by someone at Amazon, Apple or Google. Alexa could only do cricket scores because someone at Amazon built a cricket scores module. Those answers were turned back into speech by machine learning, but the answers themselves had to be created by hand. Machine learning could do the input, but not the output.

LLMs solve this, theoretically, because, theoretically, you can now not just ask anything but get an answer to anything.

Like all machine learning, LLMs turn a logic problem into a statistics problem: instead of people writing the pattern for each possible question by hand, which doesn't scale, you give the machine a meaningful sample of *all* the text and data that there is and it works out the patterns for itself, and that does scale (or should do). You get the machine to make the machine, and now you have both the input and the output. The first stage is that an LLM can answer questions about anything in that training data, but the second step is that really, this is in some senses a reasoning engine, and so you can ask it to read some web pages, or feed it some data, and find out the answer. Here's a query about a live news story in the UK this week: a topic that is not, in principle, in the training data. ChatGPT read some news websites and worked out the answer.

This is understandably intoxicating, but I think it brings us to two new problems - a science problem and a product problem. You can ask anything and the system will *try* to answer, but it might be wrong; and, even if it answers correctly, an answer might not be the right way to achieve your aim. That might be the bigger problem.

First, the science problem. We've all spent the last year talking about 'error rates' or 'hallucinations' (indeed, I also wrote about them here). The breakthrough of LLMs is to create a statistical model that can be built by machine at huge scale, instead of a deterministic model that (today) must be built by hand and doesn't scale. This is why they work, but it's inherent in this that a statistical, probabilistic model does not give a 'right' answer in a binary sense, only a probable sense. These screenshots are my favourite illustration: both of these are great results if you want a biography of someone *like* me. It gives the right *kind* of university and the right *kind* of jobs. But if you want to know which university I *actually* went to, you cannot today use ChatGPT - it will give the right 'kind' of answer, which may or may not be the 'right' answer. This is not a database.

Whether and how this can be addressed, fixed or managed is one of the basic primary science questions in AI at the moment (try those queries now and you'll get a different result, but not necessarily a better one), and you could spend a week of your life watching YouTube videos of veteran AI scientists arguing about it and conclude only that they don't really know. (It's also, in part, the AGI question, of which one could say the same.)

In the meantime, where and how does this matter? You could ask Alexa anything, but it could only answer ten things. ChatGPT will answer anything, but can you use the answer? It depends on the question.

If for some reason you really did want to know where Benedict Evans went to university, those screenshots would be useless and you should use Google. But if you were brainstorming for a biography of a character in a novel or a movie, this might be perfect. Some questions don't have wrong answers, or have a wide range of possible right answers. Meanwhile, if someone asked me for a long biography of myself and I didn't want to spend half an hour writing it, this would be great - I can see the errors and fix them, and it's still very helpful. I always used to describe the last wave of machine learning as giving you infinite interns, and that applies here: ChatGPT is an intern that can write a first draft, or a hundred first drafts, but you'll have to check it.

This is a science question, and a use-case question, but it's also a product question - how do you present and package uncertainty? This is a very basic problem in using LLMs for general web search: Google gives you Ten Blue Links, which communicates "it might be one of these - see what you think" (and turns us all into Mechanical Turks, giving Google feedback by picking the best answer). But a chat bot gives you three paragraphs of text with apparent certainty as The Answer, and footnotes, a click-through disclaimer and a 'be careful!' boilerplate at the end don't really solve that. That could be a product design question ('suggest more answers'), but I think this false impression of certainty also extends to the text format itself: LLMs can now do perfect natural language generation, which tends to hide the flaws in the underlying model. They generate prose that is grammatically and linguistically correct, and the fact that the prose is perfect tends to hide the weakness in what the prose is expressing. (The same problem applies to some extent if you ask for a table - it will give you sometime that looks like a table, but you'll have to check the numbers.)

However, stepping back, I think the product questions are actually much more broad than the error rate. I said earlier that I think Alexa and other voice assistants failed for two reasons, and one of them was that machine learning could not then solve for the output, only the input. That has changed. But the second reason is that I think that natural language, voice or text are not necessarily the right interface even if you were talking to an AGI, even if it was 'right', and, more fundamentally, that asking a question and getting an answer might be a narrow interface, not a general-purpose one.

The simplest challenge is what to ask. You have a text box and a prompt. What do you type in? What can you ask for? All conversations about AI these days seem to be hunts for metaphors, so, as an analyst, I think it's interesting to think about Excel. You're given an infinite grid that could do 'anything', so what do you do with it? What would you make? That might be a hard question. An LLM text prompt has a lot of this 'blank canvas' challenge, but with even less constraint.

Some of this is familiarity, or exploration, or desire paths, and some of my objection is ‘legacy thinking’. Whenever we get a new tool, we start by forcing it to fit our existing ways of working, and then over time we change the work to fit the new tool. We try to treat ChatGPT as though it was Google or a database instead of asking what it *is* useful for. How can we change the work to take advantage of this?

Excel, like a lot of modern software, tries to help. When you open it today, you don’t get a blank spreadsheet. You get ideas and suggestions.

ChatGPT is now trying to do the same thing - ‘what should I do with this?’

This takes me to a second problem, though. Excel isn't just giving suggestions - those tiles are documents, and documents are the start of a process, not an answer. You can see what you've built and what it's doing, and how far you've got. The same sense of creation as process applies to Photoshop, Ableton or Powerpoint, or even a simple text editor. The operative word is *editor* - you can edit!

Conversely, using an LLM to do anything specific is a series of questions and answers, and trial and error, not a process. You don't work on something that evolves under your hands. You create an input, which might be five words or 50, or you might attach a CSV or an image, and you press GO, and your prompt goes into a black box and *something* comes back. If that wasn't what you wanted, you go back to the prompt and try again, or tell the black box to do *something* to the third paragraph or change the tree in the image, press GO, and see what happens now. This can feel like Battleship as a user interface - you plug stuff into the prompt and wait to find out what you hit.

I don't think the solution is to buy the metaphorical 'ChatGPT for Dummies' book. That will tell you that it helps if you type 'imagine you're an ambitious VP at an ad agency!' before you ask for ideas for advertising copy, and that you can get the model to step through the reply, or use plug-ins, or a 'Code Interpreter'. But if you need a manual, it's not 'natural language' anymore. Once you start talking about 'prompt engineering', you're describing command lines - you're describing what came before GUIs, not what comes after them.

Rather, going back to the prompt itself, and going back to Excel, I'm reminded of a consultant I spoke to a long time ago who told me that half of his jobs were switching people from Excel to a database and the other half were the other way around. Just as every Unix function became a company, every one of those templates in the 'File/New' screen is a company. At a certain point, the kinds of control, tooling, suggestion and so on that you want become so specific to the task that they could become an entirely new product and an entirely new company. You *could* run a small business entirely in Excel and use VLOOKUP and a bunch of nested IF statements to manage invoices - but you could also use Quickbooks.

Equally, you could use ChatGPT with a code interpreter, some plugins and a text file full of saved prompts - but it might be better for those tasks to be unbundled into tools. This year there's been a wave of hundreds and perhaps thousands of first drafts of this - the so-called 'thin ChatGPT wrappers.' The prompt is an API call and 'prompt engineering' is the API parameters. That's what happened to command lines - the paradox of ChatGPT is that it is both a step forward beyond graphical user interfaces, because you can ask for anything, not just what's been built as a feature with a button, but also a step back, because very quickly you have to memorise a bunch of obscure incantations, much like the command lines that GUIs replaced, and remember your ideas for what you wanted to do and how you did it last week - and then you pay someone to turn those command into buttons.

Of course, one of the big questions in AI now is around that 'thin' in 'thin ChatGPT wrappers' - how thin? If you're just an API call with a GUI, you probably don't have much of a moat. But on the other hand, everyone in tech today is wrapping something else at some level. Snap uses Google Cloud for storage, but we don't call it a 'thin GCP Wrapper'. A big part of the way that the last wave of machine learning was deployed was that the basic capabilities became building blocks on the ‘hyperscalers’ (Google Cloud, AWS and Microsoft Azure) and those were then integrated into more specialised products further up the stack. A legal software company might use GCP translation and AWS sentiment analysis just as it used AWS data storage, but it used them to build something that law firms could buy, with a lot of other product around that, and a sales force and understanding of what lawyers needed, and it didn’t worry at all about competition from AWS. And then, as part of that product, it might built its own models with its own data doing things that the hyperscalers did not.

I don’t know that it’s clear yet how far LLMs will follow this pattern. It might be that there will be lots of smaller, cheaper, open source LLMs, or it might be that there will be a small number of models that are very expensive, very big and very good (and very generalised), and that reach much further up the stack than the models of the previous wave. Again, you could watch a lot of YouTubes of people debating this, but in this context I don’t think it changes the point: in a lot of cases, the general purpose system will probably be abstracted into single purpose UIs, even if those UIs are pretty thin.

There might an echo here of the repeated failure of pen computing in the last few decades: we use pens, so we think that’s right way to interact with a computer, and we think that once the hardware is perfected the rest will follow logically. And we talk to people, so surely, if we could talk to a computer, and it could talk to us, that would be better? Yet Apple now has a technically flawless pen computing model in the iPad, and how many people use the stylus to do their email? This is skeuomorphism. Maybe it’s better for the computer to be a computer, not pretend to be paper, and not pretend to be an intern? I’ve suggested that we’ll get lots of tooling to manage our interaction with that LLM black box, and *of course* all of this will get much much better, but it’s also possible that there really is a fundamental paradigm shift to go from work as process to a question and answer model in LLMs, that this remains inherent, and that this only applies to some tasks and not to others. Most people have no use for an Apple Pencil.

Going back to Excel and shifting my metaphor up a level, today ChatGPT sometimes seems more like the original PCs than like Excel (or VisiCalc). It's a general purpose technology, there's a command line, and some stuff that's theoretically magic, and a few things that are extremely useful to a few people, but we don't yet have the richness of all the software that came on top - all of the embodied use cases. Right now, ChatGPT is very useful for writing code, brainstorming marketing ideas, producing rough drafts of text, and a few other things, but for a lot of other people it looks a bit like those PCs ads of the late 1970s that promised you could use it to organise recipes or balance your cheque book - it can do anything, but what?