Basic and Simple

I had a pile of PDF invoices to turn into data. They came from different vendors, so the layouts didn't match. Some had tables, some had paragraphs, some had the totals on the left and some on the right. The kind of problem that makes you reach for something clever.

The fashionable thing to reach for, at the moment, is AI. So I tried the AI-powered entity extraction in Power BI [1]. It's a well-built tool. If you have a thousand invoices that all look alike, it works. Mine didn't all look alike, and it didn't work. I spent the afternoon coaxing it, and at the end of the afternoon I still didn't have the numbers out of the PDFs.

Then I remembered Ghostscript [2]. Ghostscript is about thirty years old. It reads PostScript and PDF, and it has a mode that just dumps the text:

gs -sDEVICE=txtwrite -o invoice.txt invoice.pdf

That was it. One line. The text came out, and once it was text I could do whatever I wanted with it. The problem I'd been trying to solve with a model that had read the whole internet turned out to be a problem you could solve with a program old enough to vote.

This is one of those experiences that is embarrassing in the moment and useful afterwards. The useful part is the generalization: new tools are easier to reach for than old ones, because old tools are invisible. Nobody blogs about Ghostscript. Nobody has a conference about it. It just sits there, on every Unix machine in the world, doing the same thing it's been doing since before most programmers were born. When you forget about it, you pay a tax — sometimes a small one, sometimes an afternoon.

The reason old tools get forgotten isn't that they stopped working. It's that the people who know them stopped being loud. Software has fashion cycles like clothing does, and the people setting the fashions are usually selling something. The tools that survive without marketing departments tend to be the ones that do one thing so well that the thing itself became boring. Boring is a compliment.

I'll give you another example. SNOBOL4 [3]. It's a pattern-matching language from the mid–1960s — older than C, older than Unix. You can still get it; Phil Budne maintains a free implementation called CSNOBOL4. Its patterns are recursive in a way regular expressions aren't, which means a SNOBOL4 program that parses some gnarly text format can be shorter and clearer than the equivalent regex. Twenty lines of SNOBOL4 will sometimes do what a hundred lines of Python with re will do, and you'll be able to read it afterwards.

Nobody is going to write you a SNOBOL4 tutorial on a VC-backed blog. That's not evidence against it. That's evidence that the people who use it are busy using it.

I don’t think the lesson here is that old is good and new is bad. That would be the opposite mistake. The new NLP tools are doing things that were impossible ten years ago, and some of my work wouldn’t exist without them. The lesson is narrower: when you have a specific, bounded problem, the best tool is often an old, specific, bounded tool. The match between problem and tool matters more than the age of the tool. AI is a general solvent. Ghostscript is a screwdriver. If the thing in front of you is a screw, you want the screwdriver.

There’s a related mistake worth naming. When a new technology is exciting, people try to use it for everything, including things the old technology already handled. This isn’t irrational — it’s how you learn the shape of the new thing. But it produces a period where the new thing is credited with everything and the old things are credited with nothing, and during that period a lot of afternoons get wasted.

There’s a newer wrinkle that makes this easier than it used to be. One reason people assume old tools are stuck in the past is that the new things can’t talk to them. That used to be true. It’s becoming less true. The Model Context Protocol [4], and the broader pattern of exposing tools to language models through a small, standard interface, means you can take a thirty-year-old command-line program and hand it to an LLM as a capability. Ghostscript becomes a tool the model can call. SNOBOL4 becomes a tool the model can call. Anything with a stable interface — which is most old software, because that’s why it survived — can be wrapped in a few lines and made available.

This matters more than it sounds. It means you don’t have to choose between the old world and the new one. You don’t have to rewrite the thirty-year-old program in Python so the model can reach it. You just expose it. The model does what models are good at — deciding what to do — and the old tool does what it’s always been good at. The wheel stays invented.

The economic consequence is that the cost of reusing old software, relative to rebuilding it, just dropped. And reuse was already the right answer most of the time. If something works, and has worked for thirty years, the base rate that it will keep working is very high. Rewriting it is a bet against that base rate. Sometimes the bet is worth making — the old tool may be genuinely unfit, or licensing may force your hand — but most of the time, the honest answer is that the rewrite happens because rewriting is more fun than wrapping. Fun is not a business case.

There’s enough genuinely new work to do that we don’t need to manufacture more by redoing the old work. Shipping sooner, with fewer bugs, using a program that has already survived a generation of use — that is not a compromise. That is the deal.

The practical habit I've ended up with is: before I reach for the shiny thing, I ask whether the problem is actually new. Extracting text from PDFs is not a new problem. Parsing structured text is not a new problem. A lot of what looks like a new problem, once you look at it steadily, is an old problem wearing new clothes. For those, the thirty-year-old tool is usually the right answer, and it will still be the right answer in another thirty years, long after whatever is fashionable this week has been forgotten.

The newer tools will earn their keep on the problems that really are new. There are enough of those. We don't need to invent more.

References

[1] Microsoft, *AI-powered entity extraction in Power BI / Power Query* — https://learn.microsoft.com/power-query/

[2] Artifex Software, *Ghostscript* — https://www.ghostscript.com/

[3] Phil Budne, *CSNOBOL4 (SNOBOL4 implementation)* — http://www.snobol4.org/

[4] Anthropic, *Model Context Protocol (MCP)* — https://modelcontextprotocol.io/

[5] Original version: Tom Winans, *Sometimes it's Basic and Simple*, AI GoPubby (Medium) — https://ai.gopubby.com/sometimes-its-basic-and-simple–7943276d2c15