Can 'reasoning' LLMs help with recs data creation?

25 February 2025 · ai, deep learning, records

A nervous tourist, glances back and forth between their phone and the street sign. They then rotate their phone 180 degrees, pauses, blink and frown. The lost traveller, flags a nearby ‘local’ (the passer-by has a dog on a lead. “Excuse me…” she squeaks, “How may I get to Tower Hill?”

“Well, that’ s a good one” ponders the dog walker, “You know…”

“Yes?” queries the tourist hopefully.

“Yeah…” A long pause ensues then, “Well I wouldn’t start from here” He states confidently.

The tourist almost visibly deflates and starts looking for an exit.

That’s often how we start off in software testing. Despite the flood of methodologies, tips on pairing, power of three-ing, backlog grooming, automating, refining and all the other …ings) We often find ourselves having to figure out and therefore ‘test’ a piece of software by using it. And that’s good. Its powerful, and effective if done right.

But, like our dog walker, we can sometimes find ourselves somewhere unfamiliar having to work back to front – and engineer inputs to test desired outputs. A classic example is reconciliation style systems. Imagine you have a data table or even file, it’s got columns, ID numbers, references and some amounts. This table of data is formatted in a particular way, something compatible with the accounting system or some other internal software.

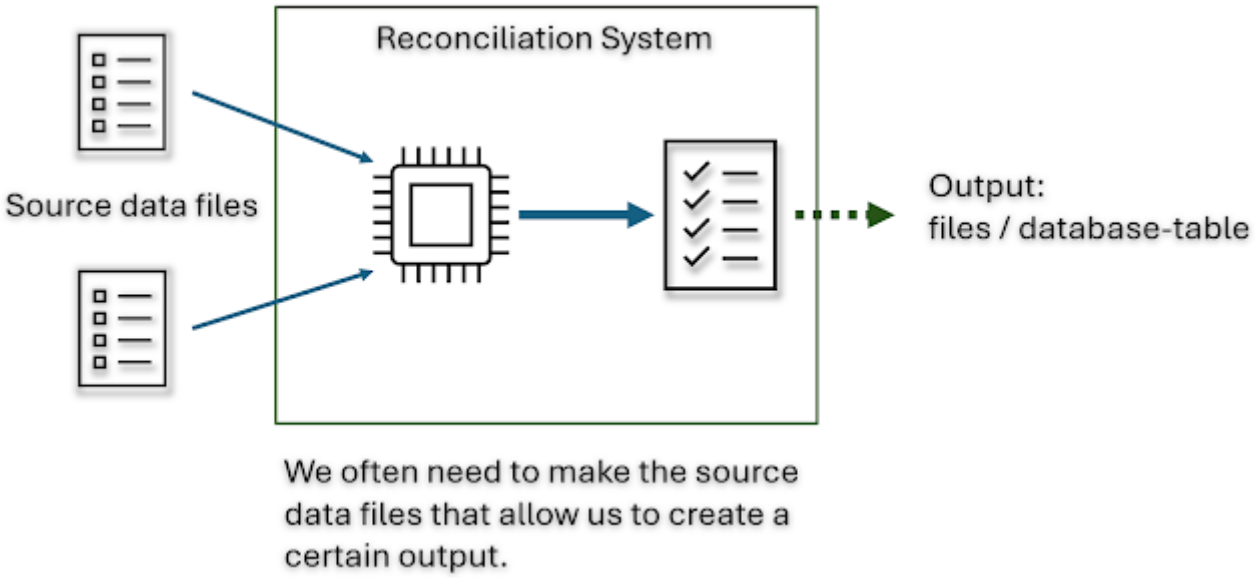
This pipe delimited file could be the file or data table we want to end up with:

|ID1|ID2|Amount|Ref|

|888|8294|100|24235|

|101|9378|205|9350|

That data table might be generated internally within a 3rd party application, and I can only create that combined table/file by importing 2 different files – that are ‘reconciled’ (joined together based on some rules…) to make the data file above.



Typically, there might be some rules that are applied, and the input files might be formats (like YAML or JSON) e.g.:

The rules for combining are as follows:

ID1 comes from the YAML file.
ID2 comes from the JSON file.
The amount is derived by adding the Amounts in both the YAML and the JSON file.
The Ref is always taken from the file that has the highest Amount.

The description is still a little vague, but the description outlines what needs to be done in these sorts of testing situations. I would probably be testing the rules that the system (under test) claims to implement, that are described above.

If reworded, that could be a problem statement that an engineer could code a solution to. Or you could generate the data manually, though for a larger dataset – this might become laborious and error prone.

Until recently Large Language Models (LLMs) also struggled a lot with this kind of problem, at least when presented in this manner. In part this is because you sort of have to think back from the output and figure out what input data would be required. LMs were predicting what's next token by token, and not able to re-assess their output for errors.

But the newer models with the 'reasoning' moniker do better. The actual 'reasoning' is that they reprocess their output and adjust as required. With these models, we can describe the problem as above in a fairly straightforward manner, it will do the iterative back and forward required to generate the input files needed.

The oft maligned chat interface works quite well, you can start with a simple definition of what you want and refine as needed. For example, the above statements don't include any description / schema of the you might need for your input YAML and JSON files. You can add this, if the one guesses at is not appropriate, and try again.

While these are not difficult problems to solve either manually or with code, those two methods sit at 2 extremes and a reasoning LLM may well provide a middle path. The ability to describe the situation you want to generate data for - and have a machine help you get a result.

This middle ground is like where Excel fits into the programming world. Many businesses are essentially based around excel. R&D departments model their data on it, for example – for products that might yield many millions of dollars. It provides a simplified way to handle and process your data, and only small bits of coding or computational thinking are usually required. Its easy enough for non-coders to use and powerful enough that it can help solve your problems.

Reasoning LLMs might fit into this realm, of enabling domain experts to fulfil tasks that previously involve an engineer. The engineer will often overthink in these situations, creating a framework – that might work well for a few examples. But the problem is better solved by a tool or toolbox, not a 'finished' test framework. Why? Well, like our tourist, each journey is different – and they might require different transport. Likewise giving the testers and domain experts these sorts of 'reasoning' tools might help them deliver ad-hoc solutions – as and when needed.

The technology has improved greatly in recent months, and I think that while the current LLMs are close – they need tailoring, prompting better to close the gap. It's going to be a useful approach to these sorts of problems.

If your interested to try for yourself, here is one of the examples (https://gist.github.com/phoughton/06c7854fde8c3b866b355e7cfb587132) I tried with ChatGPT o3-mini-high.