SUBJECT7
QUALITY IS A CULTURE

Machine Learning Hype in Testing: Separating Fact from Fiction

By Matt@xndev.com for Subject7

Three computer systems, working on the same problem separately, all try to solve a problem. If at least two come to the same conclusion, then everything can proceed. If, however, there are three different answers, then humans need to intervene to figure out what is really going on. Is this fact or fiction?

If you said this was the plot to the Tom Cruise movie *Minority Report* and fiction, then you are right. On the other hand, if you said it was the way the Mars Curiosity Rover was designed, then you are also correct. NASA used three completely independent teams to create the navigation software. If there is a defect, or a confusion about the specification, hopefully two of the three navigation systems were programmed correctly, and the controller can pick the majority report.

Like Star Trek predicted the cell phone and tablet computer, science fiction for testing is blending into science for testing today. Today I'll define AI and Machine Learning in testing, discuss what is being done now, what is hype, and how to tell the difference.

Let's start with the limits of what is possible today, to catch the hype as you hear it.

What's not possible

A few years ago at a major conference, a speaker claimed that automated tests would be able to see a failing test run, use machine learning to figure out what went wrong, fix the code, recompile, and re-run. That is, the system would self-heal the production code. I clarified that was actually his claim, said "thank you," and walked out politely. I guess I was … less polite than I intended, but it would have been a bigger problem if I stayed. I would have either supported the claim implicitly or had a rather embarrassing argument.

If you've ever debugged test failures in production, you know the "right" answer is usually unclear. The problem could be the test code, or it could be the production code. Sometimes, different people on the team disagree about what the "right" behavior is. Tickets, bugs, and stories created by humans who are fully self-aware and sentient, describing a problem well, are bounced back as "not an issue." Offhand, I'd guess 90% of customer-user-experience problems found by test tools fall into this "debatable" category that needs refinement or a conversation.

A computer isn't going to be able to solve these sorts of problems, at least, that technology is not even on the horizon today. In order to understand what a computer might do, we need to talk about what AI and ML are actually capable of right now.

AI and Machine Learning: Defined

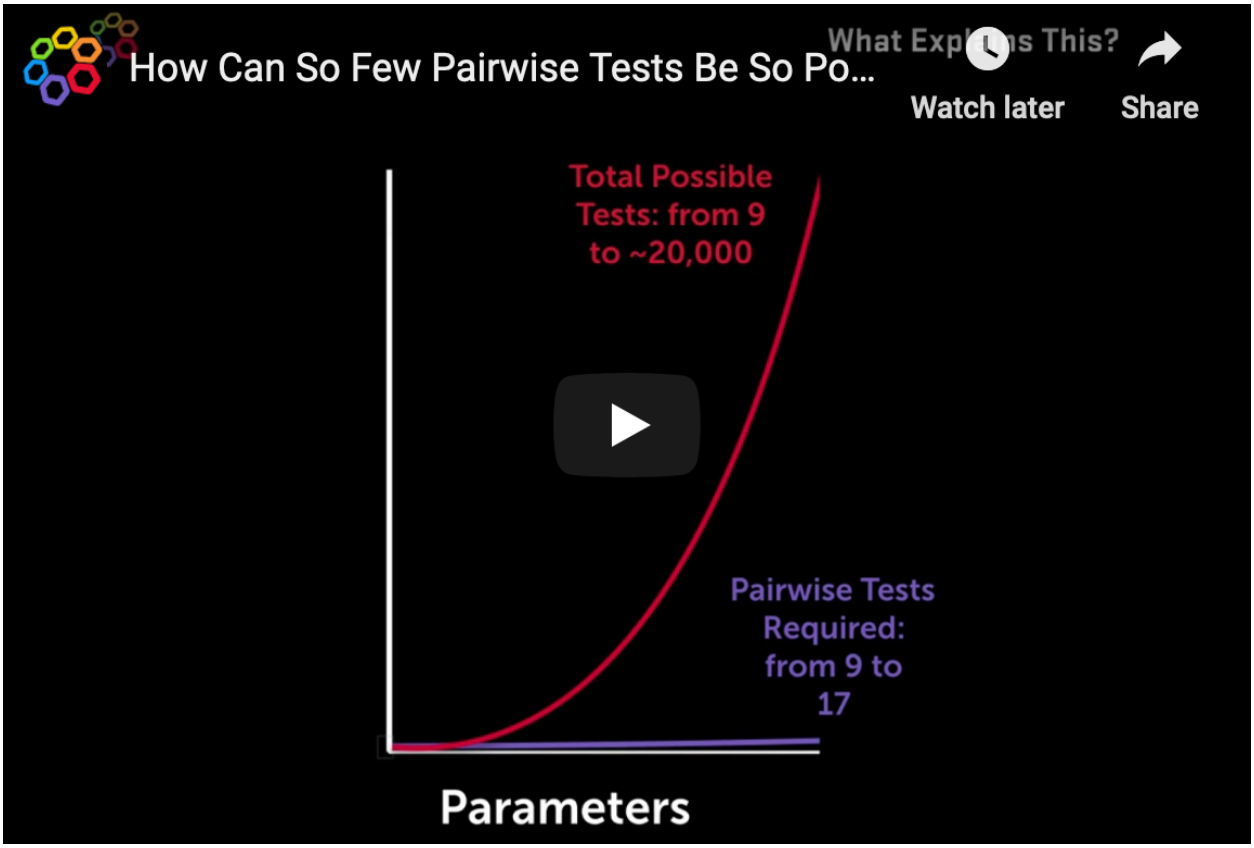
The common meaning of Artificial Intelligence today is code that can learn and make decisions. A neural net, for example, can take examples of emails that are good or spam and derive rules on which is spam. The neural net can be trained with examples, typically hundreds of thousands of examples. That is, the human that wrote the original program did not have an algorithm in mind to determine what was spam or not; the software figured that out through examples. A more IT example of ML might be trouble tickets for system operations. Say, for example, that every week, several times a week, a database operations group gets a note that a table is locked due to a race condition - two people are trying to update the table at the same time. Operations has to log in and run a command to unlock that table. There is a software company called Bigpanda that can integrate with the helpdesk tool, watch every command Operations implements to fix the broken tables, and, after some time, suggest commands to run. With some corrections, over time, you can trust BigPanda to do the work and go on to something else. Machine Learning (ML) is a subset of AI that looks at large sample sizes of data and makes guesses. If all the tickets are the same, and the sequence is a simple unlock database for (table), there might be very little data required. If the problem is larger, like inferring the rules of poker from a set of hands, you might need millions of examples. Machine learning is particularly good at prediction by data, to figure out the perfect amount of aspirin to order to keep inventory low enough without running out, based on historical data.

Artificial Intelligence in testing today

The most obvious place to use ML may be image and handwriting recognition. There are some user interfaces that do not really separate objects, but instead display a bitmap, and process the X,Y coordinates of the click. Creating test tools for this can be nearly impossible. However, Google's character recognition database has those billions of examples. Most screen fonts are rather easy to read. If test software is configured with character recognition, the tool can find the text "Submit" and click the button, making it possible to automate the user interface. That's a fantastic advancement if you are using Microsoft Terminal Server.

This is not what most of us think about when we hear the term "machine learning in testing."

Today, another use of AI is to take an incredibly large potential test set, say twenty thousand test ideas, and reduce it to the vital few, ten, or twenty. All-pairs is an approach to test design that does this. It does seem a little bit like magic, you can re-use someone else's code to do it. This is another innovation that is powerful that allows some part of the test process to move much more quickly, that is, the design of the test scenarios to execute. While this can be impressive, it is based on an algorithm, not trained with data. The computer does not "learn."



[*Click the screenshot above to be directed to the video*]

One area of ML that is advertised as available today is the so-called "self-healing" test script.

Self-Healing Test Scripts

This is possible today, albeit in a very limited sense.

When automated checks fail, it is possible they fail simply because the locator has changed. I'm not sure how often this is, it likely depends on how disciplined your team is. I would hazard a guess this is between 5 and 35% of test run "failures." It could be as simple as a textbox is now in the 3rd table row instead of the second, so /tr/td[2] needs to change to /tr/td[3] in the locator. If you always use named IDs, your percentage here might be very small.

Companies that keep their customers data in the cloud can track when there is a failure followed by a locator change followed by a pass. With enough customers and enough failures, machine learning can make an educated guess about what changes might make things work. This is doubly true if the software has access to the web page's document object model (DOM) and can try to find matches. When find_element() or click_element() fails, the software can start guessing, or, interrogate the DOM to find something else that matches. If the test passes with the change, the software can save the change, perhaps notifying people at the end of the run. Up until now I've danced around AI and ML in testing. It is time to get serious.

Where we really stand

The promise of AI today is that the new, "smarter" computer can take care of routine things, allowing the human to work more strategically. Except for test design, the examples of AI are incredibly niche. Rex Feizi, the CTO at Subject7, recently suggested to me that most, if not all of the companies working on AI for testing are focused on the self-healing locator problem, yet that is only a fraction of the automation problem. He suggested that fraction might be 5-10%. The Google character recognition tool, while impressive, is another tiny niche, solving a problem for companies still running old bitmap-serving technology. Even within the companies using locators, there isn't a great deal of evidence that they are using Machine Learning to predict fixes. From what we can tell, most companies are just traversing the tree of objects on the web page and looking for buttons to try. If companies were actually collecting data on failures and fixes, you would think the world would see a paper on common changes due to maintenance - the learnings should be spilling out into the world.

Frankly, we just don't see a lot of evidence that people are using ML for self-healing, or for anything else.

And there is a lot of room for everything else! From test data generation to environment setup, scaling, team training and reporting are all complex problems that have recurring failures. If we study the failures in large groups (with the kind of data that a Software as a Service company could get) there is real potential for software to recommend fixes. Likewise, it may be possible for AI to understand common workflows and predict test scenarios or expected results in specific domains such as eCommerce and Travel. Here's a great potential for ML: Observe real users, thousands of uses, by examining logs, then create test scenarios that are realistic and based on real use. The things that are done more often get heavier testing. We are years away from that, and even more years from a generalized solution.

The first step toward that would be just examining the logs to come up with a suggestion for how hard to test each feature. The first step toward that might be telling what percentage of the time each feature is used. That isn't even ML; it is a couple of database queries and good logging. How many of us even have that?

Again, we are years away from general solutions to test case generation.

Until then, if you hear someone touting a Machine Learning or Artificial Intelligence solution in testing, ask what it does. Then ask what algorithm they are using. Ask to see the data. If you can, figure out how you would do the same thing with a spreadsheet, and ask what makes the solution different. You probably won't walk out of the demo like I did, but if you challenge vendors to be more transparent and honest, instead of using buzzwords to sell, well … I can't see how that could possibly be a bad thing.