How AI Can Help Accelerate Testing

Chris Navrides
chris.navrides@gmail.com

Abstract

AI can beat humans at chess and Go, drive a car more safely than humans, identify cat videos better than any human; soon AI will help test code better than humans too. What is really going on at the forefront of building AI systems for test automation? This paper will walk through where AI is being applied across the software testing field, as well as practical applications being used today to test mobile apps and web pages.

Biography

Chris Navrides is the CEO and founder of Dev Tools AI which is building developer and tester centric testing tools. Prior to Dev Tools, he was the VP of Engineering at test.ai, which built the first AI based mobile & web testing solution. Before that, Chris worked at Google on automation testing for Google Play and mobile ads, and Dropbox on their mobile platform team. Chris received his Bachelors and Masters from Colorado School of Mines.

Introduction

Every day there are new applications and products with Artificial Intelligence that are being launched or talked about. There are now computers beating humans in Go and chess, driving safety, and other tasks that we thought were too complex for machines. We are also seeing computers starting to assist humans in creative applications, like Github's CoPilot for writing code. Google Engineers are even discussing bots they say are sentient.

Artificial Intelligence (AI) in the testing space is not new at this point. Over the last 5+ years AI has been a common talking point in many testing tools companies. There are many applications of AI within the testing space, such as API, microservices and data generation, to name a few. But this paper will focus mostly on applications within the UI testing space.

Background

What is AI/ML

Put simply, Artificial Intelligence (AI) and Machine Learning (ML) is where a computer program or system is not programmed for a specific set of rules about a given problem, but instead is programmed to optimize its output for a given set of inputs. It then is trained over a set of data to learn those rules.

For example, if you wanted to build a system to determine if a given object is a dog or a car. In traditional programming, you would build a set of rules like "if it has wheels and doors, it is a car, else it is a dog". With machine learning you would structure the data and let it learn those rules by training. You would give it a bunch of cars and dogs, and let it guess what it thinks it is. If it is right, it gets a positive reward, and that behavior is reinforced. Eventually it builds its own set of rules that allow it to guess if a given object is a dog or a car without the programmer needing to write

There are many types of AI and ML out there. The example above shows the power of predictive AI. This is applicable today in many apps that you use frequently from product suggestions from Amazon & Netflix, to search suggestions from Google. There are also generative AI systems that are used to create objects. Examples of these are systems like MidJourney that can make art images from a text prompt. Finally, there is reinforcement learning. These are the systems that are used to do complex actions such as driving and playing video games.

Predictive AI

This is the most common type of AI/ML systems people are used to. These systems are used across many industries and professions. They rely on having large amounts of data that is labeled correctly.

This type of model in the testing space can be used to determine if a given element is a shopping cart or a search icon. It is also used for things like Optical Character recognition to look at an image and read the characters within it.

Generative AI

Generative AI is a new and growing field. It is the process of using an AI/ML model to make new data. An example of a generative model of this type is OpenAI's GPT-3[1] that can generate fake news articles.

There are two ways that these models are trained. The first way is where they use existing training samples, say articles and text blocks, and the first part is given to the model. It is then scored based upon how well it generates the second part.

The second way models like this can be trained is to have models compete against each other. You have one model generate fake data, and then another model tries to detect if the data is real or fake. The generative model is rewarded when it "tricks" the detector model, and the detector model is rewarded when it picks real data. This type of training is called Generative Adversarial Networks (GANs) [2].

GANs can be used in the testing space to generate test data, such as API requests. It can also be used to generate test suites and test cases [3]. It is a powerful way to quickly create test data and coverage.

Reinforcement Learning

Reinforcement learning is a process where you build a model that will learn abstract concepts. Where the above examples are scored based upon "correctness" to a known solution, reinforcement learning can be applied to unknown domains.

The simplest example of this you can imagine is a robot exploring a maze. There are many paths that it can take, and it needs to find the optimal one. However, it can't see the full maze, so it must explore and try to reach the end with the quickest amount of time. It will eventually reach the end, but it then has to optimize the path. That path can include large optimizations from the original path that it found, or the potential for the original path to be the best.

This type of system is applied in many complex systems that you see today like self-driving cars or playing video games [4] or even testing the Windows operating system [5]. In the example of self-driving cars, the system itself is given a general reward (ex: get from point a to point b) and then it optimizes that reward for shortest distance, and least obstacles, like humans, hit.

This can be applied in testing domain for finding the shortest path within a given application to get to a particular screen. That system can then learn all the paths to say a settings page and use that instead of an end user writing a hard coded script for the exact steps.

UI Testing fundamentals

Within User Interface (UI) testing there are two core components that are at the heart of it: locating objects on the page and asserting against the system. All frameworks, for both mobile and web, and all other types of UI testing (ex: Roku, Xbox, etc.) rely on these two core concepts.

Locators

Locators, or selectors, are used to find an individual or a set of elements on a given page. This is done by building a set of filters against all of the elements on the page, to narrow it down until there is only the desired set of elements. For example, if you were on a login page and wanted to find the username text box, you would build a filter that looks for a text box that has name username. If you were to just look for a text box, then you would also get the password box which you don't want.

Assertions

Assertions can be done on any part of a given application; against the page or the backend or even the properties of the application such as the log file. The nature of these assertions can be derived from the needs of the business.

Most assertions are done to make sure a given object is visible or not. However, there are lots of other types of assertions such as making sure a given object is the right size, or in the right place on the screen. Assertions can even be used to make sure it is exactly correct down to the individual pixel.

Additional assertions are often useful as they can tell a more holistic view of the application under test. These can include verifying network logs were properly formed and sent or that there were no errors within the console log.

Problem

When writing UI automation, the test is written for the state of the app as it was in that particular moment in time. All subsequent changes to the app or page after that moment, could result in potentially broken or flaky tests. Changes can occur for a variety of reasons:

- Refactorings, such as changing the class names to be more appropriate
- Changing of web frameworks, i.e.; asp.net app being re-written to use react
- Random experiments that can alter the UI or backend to change the user experience
- Backend data changes that can result in the UI test assumptions being challenged

All of these changes cause instability in the underlying UI tests that must then be fixed and maintained. These issues are usually grouped into one of two types of errors: The page model/DOM changes, or changes to the flow (or test steps) the test needs to take in order to complete the desired scenario.

DOM Changes

Document Object Model (DOM) changes occur when some value within the page is modified. These can be minor, such as a CSS class changing, or major where entire elements disappear. When these happen in a given test case that is looking for them, it results in a failure; even when visually the object is still there.

Flow changes

Flow changes occur when a different set of steps appear in the application being tested. This is a common practice for things like seasonal promotions or alerting users. Many free apps will randomly show ads to a user that can take over the entire screen. International companies where payments and security requirements of specific regions have additional demands. If a test is meant to execute 100% of the time, in all these scenarios, this presents a difficult challenge.

Using AI /ML for UI Testing

Element Selection

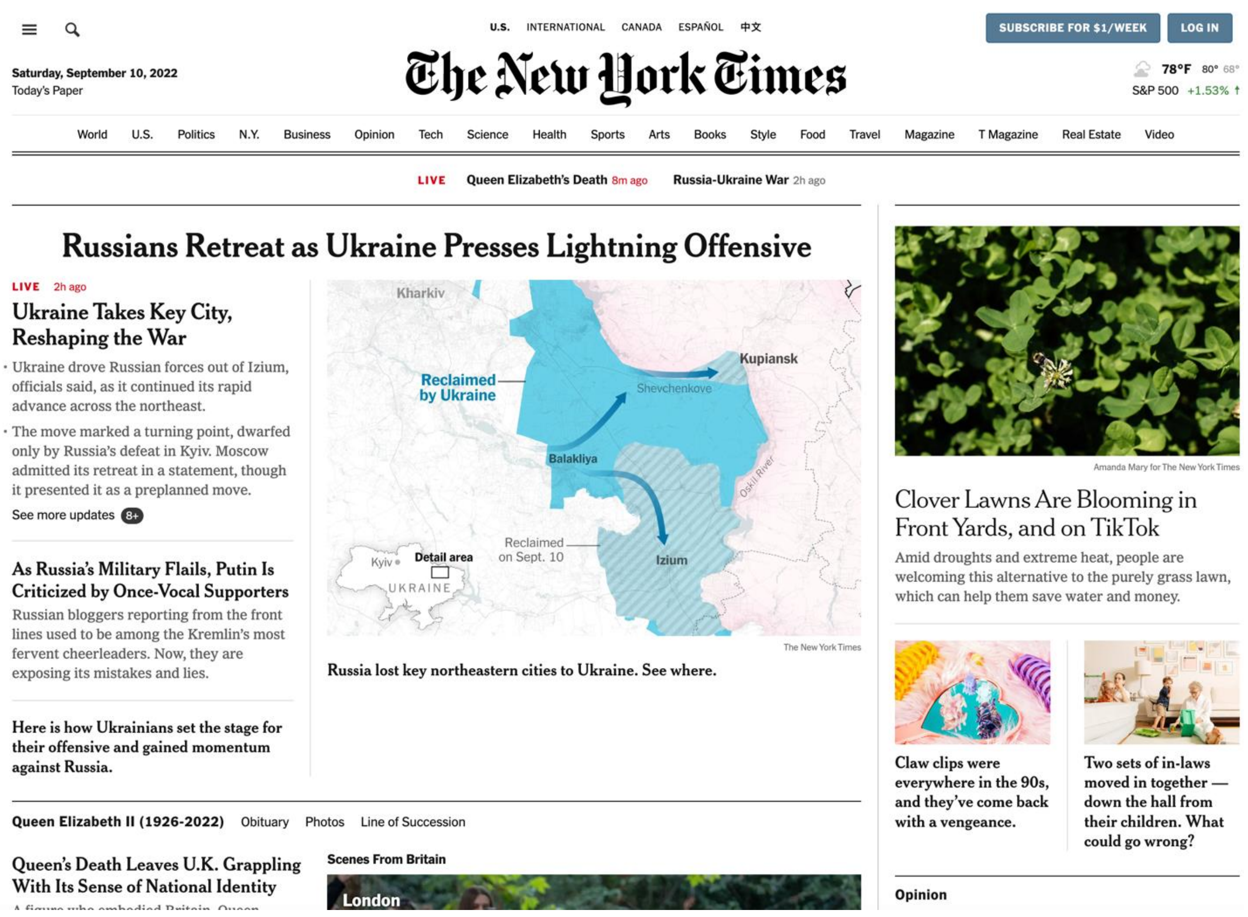
One potential solution to building locators within a UI testing framework is to use visual AI to find the element on the screen instead of using the DOM to find the element. This has several advantages in that it allows the test case to execute like an end user, and not have false negatives when an underlying change occurs. It also would be able to be executed across different platforms, such as iOS/Android and web as most applications tend to use the same iconography for all use cases.

An approach to detecting a specific element, on a given screenshot, would be to look at the screenshot and visually locate all elements on that screen. That would then enable an iterative approach to look over each element individually and determine what it is (ex: menu icon, logo, search button, etc.).

Object Detection

To detect elements on a given screen, we need to build a model. There are several model types that are valid such as R-CNN [8], You Only Look Once (YOLO) [9] and Single Shot Detectors [10]. These can be taken from reference examples or trained specifically for a given task.

What this looks like in practice is a set of bounding boxes for a given screenshot (fig.1) that the model detects. The model outputs boxes of what it sees are given elements (fig. 2).



U.S. INTERNATIONAL CANADA ESPAÑOL 中文
SUBSCRIBE FOR $1/WEEK
LOG IN
The New York Times
World U.S. Politics N.Y. Business Opinion Tech Science Health Sports Arts Books Style Food Travel Magazine T Magazine Real Estate Video
Kharkiv
Reclaimed by Ukraine
Kupiansk
Shevchenkove
Balakliya
Oskil River
Reclaimed on Sept. 10
Izium
Kyiv
Detail area
UKRAINE

Saturday, September 10, 2022
Today's Paper

78°F 80° 68°
S&P 500 +1.53% ↑

LIVE Queen Elizabeth's Death 8m ago Russia-Ukraine War 2h ago

Russians Retreat as Ukraine Presses Lightning Offensive

LIVE 2h ago
Ukraine Takes Key City, Reshaping the War

- Ukraine drove Russian forces out of Izium, officials said, as it continued its rapid advance across the northeast.
- The move marked a turning point, dwarfed only by Russia's defeat in Kyiv. Moscow admitted its retreat in a statement, though it presented it as a preplanned move.

See more updates 8+

As Russia's Military Flails, Putin Is Criticized by Once-Vocal Supporters
Russian bloggers reporting from the front lines used to be among the Kremlin's most fervent cheerleaders. Now, they are exposing its mistakes and lies.

Here is how Ukrainians set the stage for their offensive and gained momentum against Russia.

The New York Times
Russia lost key northeastern cities to Ukraine. See where.

Amanda Mary for The New York Times
Clover Lawns Are Blooming in Front Yards, and on TikTok
Amid droughts and extreme heat, people are welcoming this alternative to the purely grass lawn, which can help them save water and money.

Claw clips were everywhere in the 90s, and they've come back with a vengeance.

Two sets of in-laws moved in together — down the hall from their children. What could go wrong?

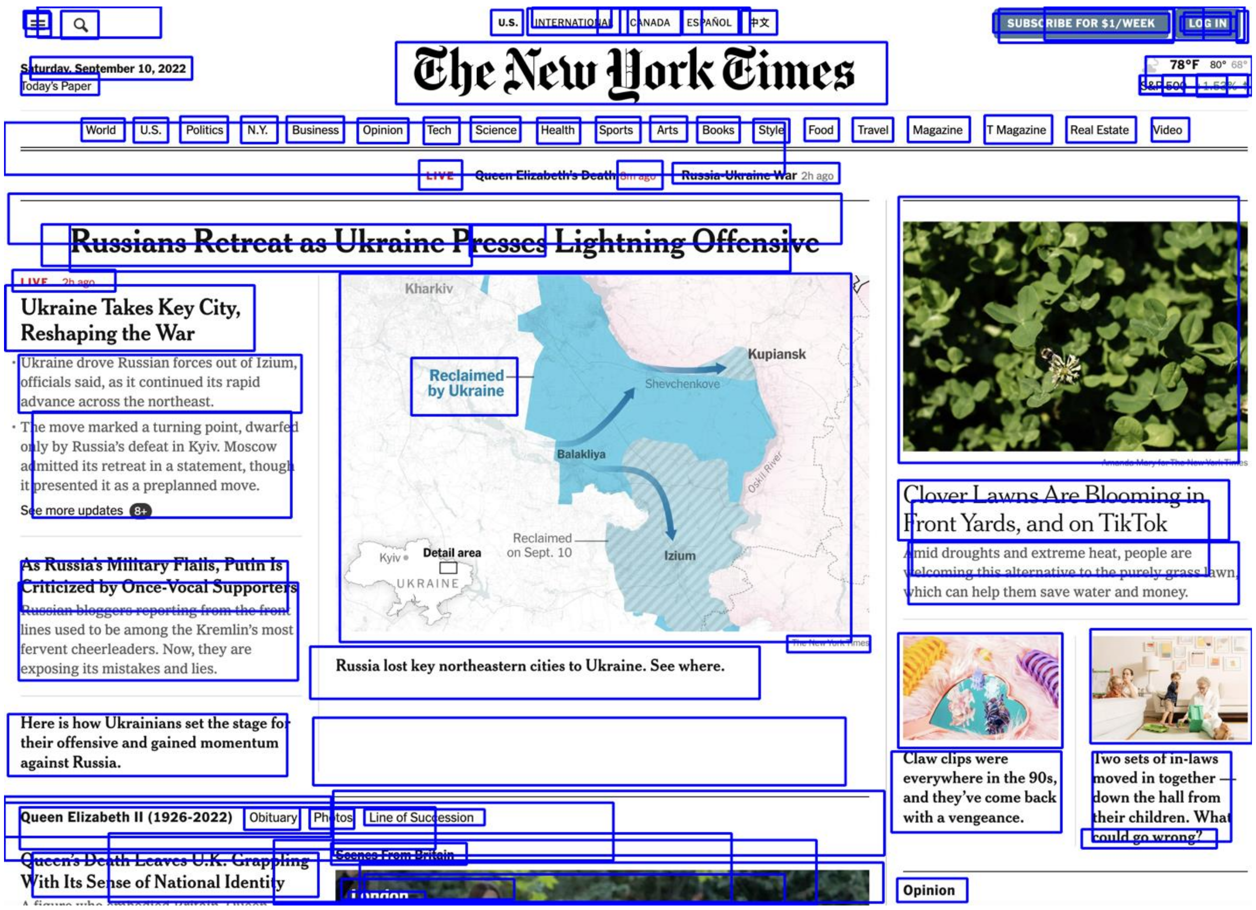
Queen Elizabeth II (1926-2022) Obituary Photos Line of Succession

Queen's Death Leaves U.K. Grappling With Its Sense of National Identity

Scenes From Britain
London

Opinion

fig 1. A reference image of the website https://nytimes.com


U.S. INTERNATIONAL CANADA ESPAÑOL 中文

SUBSCRIBE FOR $1/WEEK LOG IN

Saturday, September 10, 2022
Today's Paper

The New York Times

78°F 80° 68°
S&P 500 1.53%

World U.S. Politics N.Y. Business Opinion Tech Science Health Sports Arts Books Style Food Travel Magazine T Magazine Real Estate Video

LIVE Queen Elizabeth's Death 6h ago Russia-Ukraine War 2h ago

Russians Retreat as Ukraine Presses Lightning Offensive

LIVE 2h ago

Ukraine Takes Key City, Reshaping the War

- Ukraine drove Russian forces out of Izium, officials said, as it continued its rapid advance across the northeast.
- The move marked a turning point, dwarfed only by Russia's defeat in Kyiv. Moscow admitted its retreat in a statement, though it presented it as a preplanned move.

See more updates 8+

As Russia's Military Flails, Putin Is Criticized by Once-Vocal Supporters

Russian bloggers reporting from the front lines used to be among the Kremlin's most fervent cheerleaders. Now, they are exposing its mistakes and lies.

Here is how Ukrainians set the stage for their offensive and gained momentum against Russia.

Russia lost key northeastern cities to Ukraine. See where.

Amanda Mary for The New York Times

Clover Lawns Are Blooming in Front Yards, and on TikTok

Amid droughts and extreme heat, people are welcoming this alternative to the purely grass lawn, which can help them save water and money.

Claw clips were everywhere in the 90s, and they've come back with a vengeance.

Two sets of in-laws moved in together — down the hall from their children. What could go wrong?

Queen Elizabeth II (1926-2022) Obituary Photos Line of Succession

Queen's Death Leaves U.K. Grappling With Its Sense of National Identity

Scenes From Britain

London

Opinion

Fig 2. Element bounding boxes over the screenshot

Image Classification

The individual elements that are taken from the object detection network can then be passed to an image classification system. This system can be trained with the application specific data to better classify a given element.

This approach is agnostic to the underlying rendering technology, such as React / ASP.NET/ Kotlin / Swift/ etc. It looks purely at a given image, and what elements are there, seeing them as an end user. It gives the advantage too of not needing to understand the underlying page DOM properties.

Navigation

To assist in flow changes, tests can use reinforcement learning within a given application. This will give them the ability to robustly execute a given test, without needing to build the additional logic to check for possible new screens/pop-ups.

App Graph

To start with this approach, an understanding of the application is first needed. This can be accomplished by learning from existing tests, or by doing a crawl. The purpose of this is to understand which buttons on a given screen result in going to a new a screen.

You can determine which screen is which by using the URL or page name for each screen. Each element can be determined via it’s XPath or element name. When you combine these, you will get a graph where nodes represent a screen and edges are elements.

Q-Learn

Upon building the app graph you can apply a reinforcement learning approach. A popular reinforcement learning technique is Q-Learn. Q-Learn is an algorithm that seeks to find the best action to take given the current state [11].

This state-action graph can then be applied where for a given element you wish to interact with, you know which state/node it should be in. For each action in a given test script, a system can then look at what the current state is, and the set of actions to get to a state where the element exists. It then changes the nature of a test script to only list what actions it should execute that are pertinent to the core logic being tested by that test case.

Additional Uses AI/ML in Testing Space

Within the testing domain, AI & ML is being applied with great effect. When used effectively it can help reduce the amount of time spent on testing, while improving test coverage and product quality. These are just a few examples of testing within the space.

Test Selection

Running a large test suite can take a long time and cost a large amount of money. This is a compounding problem as a team grows. There will be more people adding code, which means more tests will be written, and those tests will be run more frequently. Eventually running every test, for every change, becomes incredibly expensive.

This was the challenge at Google. In 2014 a team was formed to see if machine learning could be applied to analyze a given change set, and only run a subset of tests [7]. The results they found were that they could reduce the number of tests executed by 25% while still obtaining high test coverage numbers.

Mutation Testing

Mutation Testing is the software testing approach where values within the object under test are randomly mutated or changed (ex: Flag turned from True → False). The test suite is run to determine if that mutation is detected. It is useful as an approach to evaluate a test framework and see the systems recovery mechanisms.

The problem with this is that there are too many permutations to try to make it effective. Machine learning is being applied to learn which mutations to optimize for and reduce the number of mutations to run [6]. This will bring the cost and time to run this down which will make it a more valuable tool for everyone.

Summary

AI & ML is being used for testing today. It has real applications at companies both big and small. There are a variety of different methods that can be used, and various areas that it can be applied to.

References

[1] https://towardsdatascience.com/creating-fake-news-with-openais-language-models-368e01a698a3
[2] https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/
[3] https://arxiv.org/abs/2104.11069
[4] https://www.technologyreview.com/2022/05/27/1052826/ai-reinforcement-learning-self-driving-cars-autonomous-vehicles-wayve-waabi-cruise/
[5] https://arxiv.org/pdf/2007.08220.pdf
[6] https://link.springer.com/content/pdf/10.1007/978-3-642-34691-0_15.pdf
[7] https://testing.googleblog.com/2018/09/
[8] https://medium.com/coinmonks/review-r-cnn-object-detection-b476aba290d1
[9] https://iopscience.iop.org/article/10.1088/1742-6596/1004/1/012029/pdf

[10] https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-020-01826-x
[11] https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56