Parasoft Blog

Was bedeutet künstliche Intelligenz bei Software-Tests für IoT?

Autor: Igor Kirilenko, VP Development bei Parasoft, zeichnet für die technische Strategie, Architektur und Entwicklung von Parasofts Produkten verantwortlich.

Der Begriff „Künstliche Intelligenz“ (KI) gehört derzeit zu den aktuellsten Schlagwörtern der Softwareentwicklung. Was ist damit aber genau gemeint und wie unterscheidet sich die KI im Kontext von automatisierten Softwaretests von ihrer breiter gefassten Definition? Was bedeuten KI und der damit verwandte Begriff „Machine Learning“?

Implementierung von KI in der Testautomatisierung

Künstliche Intelligenz ist einer der am meisten überladenen Begriffe auf dem digitalen Markt. Ist von KI die Rede, gibt es mitunter wilde Assoziationen wie etwa allmächtige Supercomputer, die wild entschlossen sind, die Menschheit zu zerstören, sprachgesteuerte Assistenten in der Art von Alexa oder Siri, Computer als Schachgegner oder selbstfahrende Autos.

Die englische Wikipedia definiert künstliche Intelligenz als die Lehre von „intelligenten Akteuren“: Darunter seien Apparate zu verstehen, die ihre Umgebung wahrnehmen und Aktionen ausführen, um ihre Chance zum erfolgreichen Erreichen ihrer Ziele zu maximieren. Das ist natürlich eine sehr abstrakte Definition. Ich dagegen betrachte KI lieber als die Fähigkeit eines Computerprogramms oder einer Maschine, zu denken (d. h. eigenständig Schlussfolgerungen zu treffen) und zu lernen (d. h. Daten zu sammeln und das eigene künftige Verhalten auf vorteilhafte Weise zu modifizieren). In dieser Definition zeichnet sich für uns etwas Aussagefähigeres im Kontext dessen ab, was KI für die Softwareentwicklungs-Tools und die entsprechende Technologie bedeutet.

Wichtig ist auch sich klarzumachen, dass sich der Umfang dessen, was man unter KI versteht, mit der Zeit verändert. Beispielsweise gab es eine Zeit, in der die optische Zeichenerkennung (Optical Character Recognition, OCR) als der Stand der Technik im Bereich der KI angesehen wurde. Auch die Fragen und Antworten von Siri und Alexa galten einmal als der neueste Stand der Technik, werden inzwischen aber weitgehend als Selbstverständlichkeit hingenommen und nicht in jedem Fall als KI betrachtet. Dasselbe wird bei den Softwaretest-Tools passieren: Heutige Innovationen im Bereich der Automatisierung werden zur Selbstverständlichkeit, sobald neue Fähigkeiten bereitstehen.

AI in der Softwaretest-Automatisierung 2020

Noch steckt die Nutzung von KI in der Softwareentwicklung in den Kinderschuhen, der Autonomiegrad ist hier noch weit niedriger als auf weiter entwickelten Gebieten wie den selbstfahrenden Systemen oder der Sprachsteuerung. Dennoch zielt die Entwicklung ganz klar in Richtung des autonomen Testens. Der KI-Einsatz bei den Softwaretest-Werkzeugen zielt auf die Vereinfachung des Softwareentwicklungs-Lebenszyklus‘ ab. Durch die Anwendung von Schlussfolgerung, Problemlösung und in einigen Fällen auch maschinellem Lernen lässt sich KI nutzen, um bei der Automatisierung zu helfen und den Aufwand an mühseligen Routinearbeiten beim Entwickeln und Testen von Software zu verringern. An dieser Stelle wäre die Frage angebracht, ob Testautomatisierungs-Tools das nicht bereits tun. Die Antwort ist ein klares ja, wenn auch mit ein paar Einschränkungen
KI kann in der Softwareentwicklung immer dann glänzen, wenn es darum geht, diese Einschränkungen zu beseitigen, damit die Tools zum Automatisieren von Softwaretests den Entwicklern und Testern einen noch größeren Nutzen bieten. Der Nutzen der KI resultiert daraus, dass die direkte Einbeziehung der Entwickler oder Tester in die besonders profanen Aufgaben reduziert wird. (Auch wenn die menschliche Intelligenz nach wie vor u.a. bei der Anwendung von Geschäftslogik dringend notwendig ist.)

Bedenken Sie beispielsweise, dass die meisten (wenn nicht alle) Testautomatisierungs-Tools dafür genutzt werden, Tests für Entwickler durchzuführen und Ergebnisse zu liefern. Die meisten Tools wissen nicht, welche Tests sie durchführen sollen, also führen sie alle oder einen vorgegebenen Satz durch. Wie wäre es also, wenn ein KI-fähiger Bot den aktuellen Teststatus, neuere Codeänderungen, die Codeüberdeckung und andere Metriken auswerten könnte, um auf dieser Basis zu entscheiden, welche Tests noch notwendig sind, und diese Tests anschließend auszuführen? Das Einbringen einer Entscheidungsfindung auf der Basis veränderlicher Daten ist ein Beispiel für den Einsatz von KI. Damit ist die Software effektiv in der Lage, den Entwickler bzw. Tester in diesem Entscheidungsprozess zu ersetzen. Die Vorteile, die dies in einer CI/CD-Pipeline hat, liegen auf der Hand.

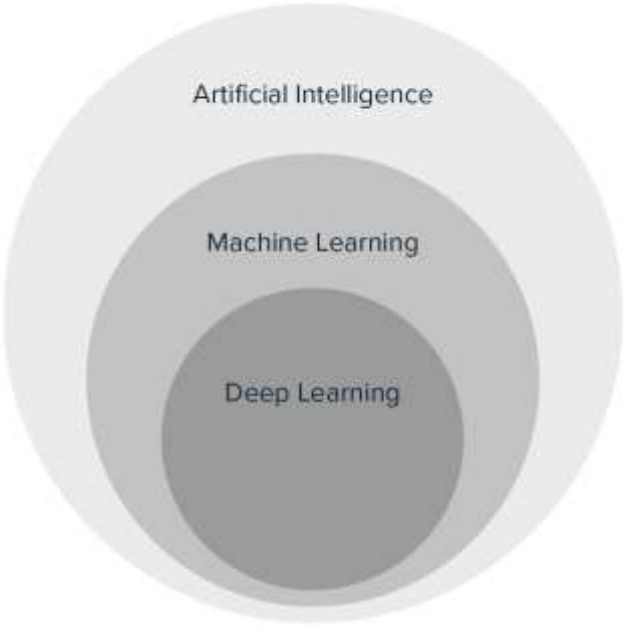
KI und maschinelles Lernen

Wie sieht es mit dem maschinellen Lernen (ML) aus? Es kann die KI durch Algorithmen aufwerten, die es dem Tool ermöglichen, automatisch besser zu werden, indem es die durch das Testen produzierte Fülle an Daten sammelt.
In anderen Fällen ist das Erfassen von Daten der Schlüssel zur Entscheidungsfindung, und maschinelles Lernen kann extrem wertvoll sein, da es zunächst einige Daten benötigt und sich dann durch die Sammlung von mehr Daten steigert beziehungsweise anpasst. So können z. B. Codeabdeckung, Ergebnisse der statischen Analyse, Testergebnisse oder

andere Software-Metriken im Laufe der Zeit die KI über den Zustand des Softwareprojekts informieren.

Deep Learning

Maschinelles Lernen hat seine eigene Untergruppe 'Deep Learning', die auf der Verarbeitung riesiger Datenmengen basiert, um daraus zu lernen. Solche Daten werden in den meisten Fällen durch mehrschichtige analytische neuronale Netze dargestellt - das sind Modelle, die vom Netz menschlicher Neuronen inspiriert sind und Computern helfen, neues Wissen zu erwerben und hochintelligent zu denken.



Der Schlüsselaspekt von Deep Learning ist die riesige Menge an Informationen, die von neuronalen Netzwerken dargestellt werden, um den Entscheidungsprozess voranzutreiben. Eine solche Datenmenge ist nicht immer verfügbar oder im Software-Testing nicht anwendbar - vielleicht ist das der Grund, warum wir noch nicht viele Anwendungsfälle von Deep Learning in diesen Bereichen sehen. Ein mögliches Beispiel wäre das "Lernen" aus zig Millionen Codezeilen, um verschiedene Arten von Sicherheitsverletzungen zu verstehen und eine statische Analyse-Engine zu implementieren, die auf einem Deep Learning-Modell basiert.

Echte Beispiele für KI und ML beim Software-Testen

Dies ist ein wichtiger Bereich der Forschung und Entwicklung bei Parasoft. Spannenderweise sind unsere aktuellen Angebote nur der Anfang und unsere laufende Forschung im Bereich KI und ML bringt immer wieder neue Möglichkeiten, diese

Technologien in unsere Produkte zu integrieren. Hier sind ein paar Möglichkeiten, wie wir sie bereits eingebracht haben.

Einsatz von künstlicher Intelligenz zur Automatisierung der Unit-Test-Generierung und Parametrisierung

Dieses erste Beispiel findet sich in Parasoft Jtest, unserer Softwaretestlösung für Java-Entwickler, die u.a. statische Analyse, Unit-Tests, Abdeckung und Nachvollziehbarkeit beinhaltet. Durch den Einsatz von KI haben wir hier eine automatische Testfallgenerierung freigegeben, die Entwicklern dabei unterstützt, die Lücken zu füllen, wenn sie von einem spärlichen JUnit-Harness ausgehen.

Das IDE-Plug-in von Parasoft Jtest bereichert die Modultests durch nützliche Informationen, indem es einfache, mit einem Klick aktivierbare Aktionen zum Erstellen, Skalieren und Pflegen von Modultests bereitstellt. Durch den Einsatz von KI-fähigem Jtest können Benutzer eine höhere Codeabdeckung erreichen und gleichzeitig den Zeit- und Arbeitsaufwand für die Erstellung einer umfassenden und aussagekräftigen Suite von JUnit-Testfällen halbieren. Eine Möglichkeit, dies zu erreichen, besteht darin, das Erstellen von Stubs und Mocks zur Isolierung des zu testenden Codes zu erleichtern. Mit der zugrunde liegenden KI kann Jtest das zu testende Modul beobachten, um seine Abhängigkeiten von anderen Klassen zu ermitteln. Werden Instanzen dieser Abhängigkeiten erstellt, schlägt es dem Benutzer vor, sie zu mocken, um besser isolierte Tests zu erstellen. Das automatische Erstellen der notwendigen Mocks und Stubs reduziert den Aufwand für einen der zeitaufwändigsten Teile der Testerstellung.

Parasoft Jtest erkennt auch automatisch Code, der von bestehenden Testsuiten nicht abgedeckt wird und durchläuft den gesamten Steuerpfad des Quellcodes, um herauszufinden, welche Parameter an eine zu prüfende Methode übergeben werden müssen, und wie Subs/Mocks zu initialisieren sind, damit sie diesen Code erreichen. Durch die Aktivierung dieser KI kann Jtest automatisch neue Modultests generieren und dabei modifizierte Parameter anwenden, um die Codeabdeckung des gesamten Projekts zu erhöhen.

Automatisiertes Generieren und Pflegen von API-Tests mit KI und ML

Ein weiteres gutes Beispiel, das Maschinelles Lernen in den Mix einbringt, ist der Smart API Test Generator von Parasoft SOAtest, der über das Aufzeichnen und Abspielen von Tests hinausgeht und mithilfe von KI und ML manuelle UI-Tests in vollständige, automatisierte API-Testszenarien umwandelt. Der Smart API Test Generator nutzt Schlussfolgerungen, um die Muster und Beziehungen in den verschiedenen API-Aufrufen zu verstehen, die bei der

Ausübung der Benutzeroberfläche erfolgen. Aus dieser Analyse wird eine Reihe von API-Aufrufen erstellt, die die zugrunde liegenden Schnittstellenaufrufe während des UI-Flows repräsentieren. Anschließend kommt das Maschinelle Lernen ins Spiel, indem so viel wie möglich über die verschiedenen API-Ressourcen beobachtet und als Vorlage in einer proprietären Datenstruktur hinterlegt wird. Über die Untersuchung von anderen Testfällen in der Bibliothek des Anwenders erfolgt die Aktualisierung dieser internen Struktur, um verschiedene Verhaltensweisen beim Ausführen der APIs zu erlernen, wie etwa eine Assertion oder das Hinzufügen eines bestimmten Headers an der richtigen Stelle.
Ziel des KI-Einsatzes ist hier das Generieren ausgefeilterer Tests, anstatt einfach die Aktionen des Anwenders zu kopieren, wie es bei einfachen Record-and-Playback-Tests wäre. Das Tool erkennt Muster innerhalb des Datenverkehrs, erstellt ein umfassendes Datenmodell der beobachteten Parameter, generiert automatisierte API-Tests und ermöglicht die Anwendung der erlernten Muster auf andere API-Tests, um diese zu verbessern und den Anwendern beim Erstellen anspruchsvollerer automatisierter Testszenarien zu helfen. Die daraus resultierenden automatisierten API-Tests sind vollständig wiederverwendbar, skalierbar und unempfindlich gegenüber Änderungen.

ML zur Selbstheilung der Ausführung von Selenium-Tests

Die automatische Validierung der Benutzeroberfläche ist eine weitere kritische Komponente Ihrer Teststrategie, um sicherzustellen, dass das Produkt vollständig verifiziert ist, bevor es in Produktion geht. Das Selenium-Framework hat sich als klare Wahl für UI-Tests durchgesetzt, aber die Anwender kämpfen immer noch mit den üblichen Selenium-Test-Herausforderungen wie Wartbarkeit und Stabilität. Hier können KI-Technologien und insbesondere maschinelles Lernen helfen, indem sie eine Selbstheilung zur Laufzeit bieten, um die häufigen Wartbarkeitsprobleme bei der UI-Testausführung zu beheben.
Diese Funktionalität bieten wir mit Parasoft Selenic an, das während der regulären Ausführung von Selenium-Tests über Ihre interne Datenstruktur "lernen" kann. Die Selenic-Engine überwacht jeden Lauf und erfasst detaillierte Informationen über den Web-UI-Inhalt der zu testenden Anwendung. Sie extrahiert DOM-Elemente, ihre Attribute, Platzhalter usw. und korreliert sie mit Aktionen, die von UI-gesteuerten Tests ausgeführt werden. Selenic verwendet den proprietären Datenmodellierungsansatz von Parasoft und speichert diese Informationen in seiner KI-Engine. Das Modell wird kontinuierlich aktualisiert und analysiert die historische Ausführung aller Tests, um immer "intelligenter" zu werden.
Dies ist eine entscheidende Zeitersparnis in Fällen, in denen UI-Elemente von Webseiten verschoben oder signifikant verändert werden, was zum Fehlschlagen von Tests führt. Mit Selenic kann die von der Engine verwendete KI-Heuristik diese geänderten Elemente mit historischen Daten, die das Modell repräsentiert, "abgleichen" und automatisch "intelligente

Platzhalter" generieren, die resistent gegen Änderungen sind, um die Ausführung von Selenium-Tests zur Laufzeit wiederherzustellen. Diese Änderungen werden automatisch an das Modell weitergegeben und die zukünftige Generierung neuer Locators erfolgt auf Basis dieser Änderungen.

Mit KI für Softwaretests zur höheren Akzeptanz der statischen Analyse

Eines der Hindernisse für die erfolgreiche Einführung statischer Analysewerkzeuge ist die Verwaltung einer großen Anzahl von Warnungen und der Umgang mit False Positives (Warnungen, die keine echten Bugs sind) in den Ergebnissen. Software-Teams, die eine Legacy- oder bestehende Code-Basis analysieren, kämpfen mit den anfänglichen Ergebnissen, die sie mit der statischen Analyse erhalten, und sind von dieser Erfahrung so abgeschreckt, dass sie keine weiteren Anstrengungen unternehmen. Ein Grund für die Zurückhaltung sind die vielen Standards, Regeln (Checkern), Empfehlungen und Metriken, die mit modernen statischen Analysewerkzeugen möglich sind.
Software-Entwicklungsteams haben einzigartige Qualitätsanforderungen und es gibt keine allgemeingültigen Empfehlungen für Checker oder Codierungsstandards. Jedes Team hat seine eigene Definition von "falsch positiv", was oft eher "egal" als "das ist technisch falsch" bedeutet. Parasofts Lösung hierfür ist die Anwendung von KI und Machine Learning, um die von der statischen Analyse gemeldeten Befunde zu priorisieren und so die Benutzererfahrung und Akzeptanz solcher Tools zu verbessern.

Parasoft nutzt eine Methode, um die Befunde in der Ausgabe eines statischen Analysetools schnell als etwas zu kategorisieren, das das Team entweder sehen oder unterdrücken möchte, indem es eine kleine Anzahl von Befunden überprüft und einen Klassifizierer auf der Grundlage der mit diesen Befunden verbundenen Metadaten konstruiert. Dieser Klassifizierer basiert auf den Ergebnissen früherer manueller Einstufungen von Befunden der statischen Analyse im Kontext sowohl der historischen Unterdrückung irrelevanter Warnungen als auch der Priorisierung sinnvoller Befunde zur Behebung innerhalb der Codebasis.
Die Endergebnisse werden auf zwei Arten eingestuft:
- Von Interesse für das Team, um es zu untersuchen.
- Elemente, die unterdrückt werden können.

Davon profitiert die Benutzerfreundlichkeit enorm, da die Entwickler zu den Warnungen geleitet werden, die mit der höchsten Wahrscheinlichkeit auf ihr Projekt zutreffen. Mit diesen Neuerungen können Unternehmen den manuellen Aufwand bei der Einführung und Nutzung der statischen Analyse sofort reduzieren.

Die Zukunft von KI und ML

Wie geht es also weiter? Wir betreiben aktive Forschung und Entwicklung auf diesem Sektor und suchen nach weiteren Anwendungsmöglichkeiten für KI und ML, um unser Angebot an Softwaretest-Tools weiter aufzuwerten. Die Forschung kann unterschiedliche Wege gehen, doch das Endziel ist klar: Den Teams soll geholfen werden, ihren Code effektiver und effizienter zu entwickeln und zu testen und mit erhöhter Schlagzahl qualitativ hochwertigere Software hervorzubringen.