Advertorial

Wie KI das Leben der Entwickler und IT-Architekten grundlegend verändert und was sie tun können

Die Automatisierung kommt – wie behaupten Sie sich?

von Bartek Rohard Warszawski

Sie befinden sich vielleicht in Ihren Zwanzigern, Dreißigern oder Vierzigern und reiten auf der neuesten technologischen Welle, ohne sich vor dem zu fürchten, „was noch bevorsteht“. Wir wissen, dass KI (Künstliche Intelligenz) mit ihrem maschinellen Lernen und der kognitiven Automatisierung sich noch in tiefen, unbekannten Gewässern tummelt und bald den Transport- und Fastfood-Sektor sowie die Landwirtschaft revolutionieren wird. Wir wissen sogar, dass diese tiefgreifende Veränderung nicht nur das Leben der gewerblich Beschäftigten beeinflussen wird, sondern dass fast alle Berufsbilder betroffen sind, vom Buchhalter über den Rechtsanwalt bis zum praktischen Arzt und Data Scientist im Kontakt mit KI. Die Fragen, um die es hier geht, lauten: Wie wird sich die künstliche Intelligenz auf die Softwareentwicklung auswirken und somit auf die Entwickler und IT-Architekten? Und was sollten wir von der bevorstehenden Welt der 2020er-Jahre erwarten, und wie stellen wir uns darauf ein?

Um zu verstehen, was uns im kommenden Jahrzehnt bevorsteht, müssen wir betrachten, was in den vergangenen 61 Jahren im Bereich der künstlichen Intelligenz geschehen ist.

Die Geschichte der künstlichen Intelligenz

Das erste Machine-Learning-Programm und neuronale Netzwerke

Der Begriff des „maschinellen Lernens“ ist nicht neu. Das erste elektronische Gerät, das es vermochte, Muster zu erkennen und zu lernen [Cor57], wurde bereits 1957 von Frank Rosenblatt gebaut. Der erste Prototyp für maschinelles Lernen wurde 1961 von Leonard Uhr und Charles Vossler gebaut [Pat61]. Neuronale Netzwerke gibt es schon lange, ihr Lernprozess erforderte jedoch große Mengen gelabelter digitaler Daten, die in den 1960er-Jahren noch selten waren oder deren Beschaffung sehr teuer war. In der digitalen Welt von heute liefern insbesondere die sozialen Medien im Sekundentakt große Mengen gelabelter Daten. Außerdem waren Sensoren früher analog und nicht sehr genau, verglichen mit den heutigen Sensoren in Telefonen, Smarthomes und Autos, die ebenfalls Daten liefern. Die fehlende Genauigkeit ist möglicherweise der Grund, warum Schach für die Automatisierung so interessant war, da die sensorische Dateneingabe einfach war (nur 8x8 Felder und 12 Arten von Spielsteinen), jedoch zu mindestens 10^{120} möglichen Schachspielvarianten führen konnte [Che], mehr als die Anzahl der Atome in unserem Universum [Uni].

Das Schachspiel an sich war einfach zu codieren, ein Schachspiel zu gewinnen hingegen nicht. Es war offenkundig, dass neuronale Netzwerke über ein großes Potenzial verfügten, aber kein neuronales Netzwerk war zu jener Zeit in der Lage, einen menschlichen Schachweltmeister zu schlagen.

Die 2000er-Jahre

In den 2000er-Jahren ist etwas passiert, das KI voranbrachte und in die Lage versetzte, Probleme zu lösen, die sie vorher nicht zu lösen vermochte. 2004 organisierte die Defense Advanced Research Projects Agency (kurz DARPA) einen Wettbewerb zum selbstfahrenden Auto – die „DARPA Challenge". Kein Fahrzeug schaffte es über den kompletten Kurs, und die längste gefahrene Distanz belief sich gerade einmal auf 12 km.

2005 unternahm DARPA erneut einen Anlauf und erzielte dabei ein völlig anderes Ergebnis: 23 von 24 Autos schlugen den Rekord von 12 km, und 5 von 24 Fahrzeugen absolvierten den ganzen Kurs mit einer Gesamtlänge von 212 km. Was war geschehen? Die unglaublichen Verbesserungen im Zusammenhang mit Big Data, erschwinglichen Hochleistungsrechnern und DEEP-Architekturen, wie z. B. digitale Online-Karten, Bildanalysetools usw.

Deep Learning

Softwareentwickler und Wissenschaftler brauchten nun geringere Datenmengen, um beispielsweise einen Algorithmus zur Bilderkennung zu entwickeln, der als neuronales Netzwerk zum Erlernen der Bilderkennung realisiert wurde. Dies war möglich, weil in den späten 2000er-Jahren ausreichend gelabelte digitale Daten für Machine-Learning-Software verfügbar waren. So konnte der ursprüngliche Algorithmus zur Bilderkennung, den Menschen bereits entwickelt hatten, mittels eines neuronalen Netzes reproduziert werden. Eine der Methoden des maschinellen Lernens, das Deep Learning, welches das Lernen in mehrere Ebenen aufteilt, wurde hierfür verwendet. Im Deep Learning lernte die erste Ebene, einfache Pixelformen zu erkennen. Die nächste Ebene lernte, aus einfachen Formen Gesichter zu filtern. So wurde der Algorithmus reproduziert.

Der heutige Stand der Technik im Deep Learning umfasst bspw. die maschinelle Diagnose von Krankheiten auf der Grundlage von Bildern. Diese Diagnose ist besser und genauer als die entsprechende Diagnose der menschlichen Spezialisten.

Deep Learning hat also im obigen Beispiel nicht nur die vorherigen Algorithmen zur Bilderkennung neu erstellt, sondern diese sogar verbessert. Maschinen erkannten nicht mehr nur Dinge auf Bildern, sondern schickten sich an, ihre eigenen Bilder zu kreieren – „Computerträume" [Ifl]. Maschinen fingen sogar an, auf bestehenden Fotos die Jahreszeit zu verändern, also etwa von einer Sommer- in eine Winterlandschaft zu wechseln, und den Tag zur Nacht zu machen und umgekehrt [Pix]. Die Machine-Learning-Algorithmen waren nicht neu, die neuen riesigen Datensätze eröffneten indes neue Möglichkeiten.

Künstliche Intelligenz – was erwartet uns?

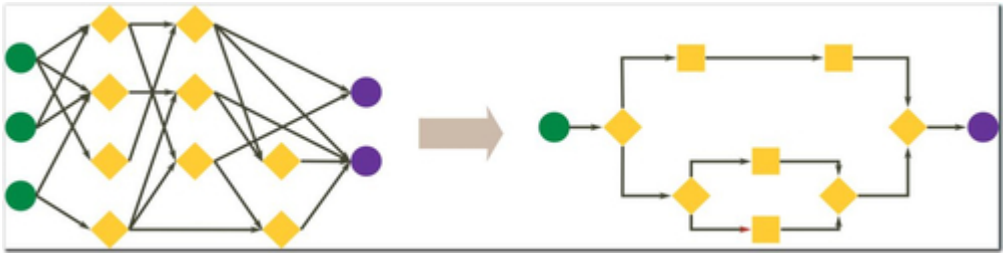

Abb. 1: Verfeinerung komplexer Muster in Algorithmen

Softwareentwicklung durch Menschen

Bei Software geht es darum, Input mit dem richtigen Output abzubilden und dadurch spezielle Probleme zu lösen. Ein Softwareentwickler oder Informatiker könnte ein komplexes Alltagsphänomen analysieren und die notwendigen Muster in einem Algorithmus aufbereiten (**siehe Abbildung 1**).

Softwareentwicklung durch Maschinen

Bei Machine Learning geht es zumeist um die Verbindung von Input und Output. Aber Machine Learning auf der Grundlage

evolutionärer Algorithmen kann sich selbst verbessern, indem es Teilstücke von Codes zusammenfügt und daraus völlig neue Algorithmen bildet. Diese Art evolutionärer Entwicklung ist viel schneller als die traditionelle Softwareentwicklung durch Menschen.

1997 gelang es Menschen, Deep Blue zu entwickeln, den ersten Computer, der in der Lage war, eine Schachpartie gegen einen menschlichen Weltmeister zu gewinnen. Es dauerte weitere 20 Jahre, bis AlphaGo 2017 in der Lage war, den weltbesten Spieler im Brettspiel Go zu besiegen.

AlphaGo Zero (eine neue Version von AlphaGo) wurde auf der Grundlage des Bestärkenden Lernens oder auch Reinforcement Learning entwickelt. AlphaGo Zero lernte in nur vier Stunden von Grund auf und selbstständig Go zu spielen und war danach in der Lage, den ursprünglichen AlphaGo-Computer hundertmal hintereinander zu schlagen [Dep].

Eine Maschine konnte nunmehr in nur vier Stunden einen besseren Algorithmus entwickeln – im Vergleich hierzu hatte eine große Anzahl an Entwicklern und Architekten zuvor 20 Jahre gebraucht. Softwareentwickler und Architekten stehen vor einer Revolution, aber die Menschen werden weiterhin gebraucht.

Es mag nicht ganz fair sein, die Entwicklungszeit der Menschen lediglich mit der Trainings- bzw. Lernzeit der Maschine zu vergleichen, da es auch seine Zeit gebraucht hat, den Trainings-/Lernalgorithmus zu entwickeln. Man darf bei dieser Überlegung aber nicht vergessen, dass der Trainings-/Lernalgorithmus für andere Projekte wiederverwendet werden kann – was bei der menschlichen Entwicklungszeit nicht der Fall ist.

Die Gefahr, nicht zu verstehen

In den vergangenen 70 Jahren wurde Software auf der Grundlage einfacher Algorithmen entwickelt, die von Menschen verstanden werden können (auch wenn die meisten sich damit schwertun). Machine-Learning-Software kann eine neue Art von Algorithmen entwickeln, die viel komplexer und für den Menschen nicht mehr nachvollziehbar sind. Wir sind heute in viel höherem Maße von unserer Software abhängig als je zuvor, und wir nehmen ein großes Risiko in Kauf, wenn wir die Algorithmen hinter unserer Software nicht mehr verstehen.

Die Folgen sind noch überschaubar, wenn ein Wetter-Chatbot das Wort „heute Abend“ für einen Städtenamen hält und die Frage „Wie ist das Wetter heute Abend?“ unzureichend mit: „Ich kenne ‚heute Abend‘ nicht. Versuchen Sie es mit einer anderen Stadt oder einem anderen Stadtteil“, beantwortet wird. Missverstandene Wörter hätten viel gravierendere Konsequenzen in einem kritischen Gesundheitssystem, insbesondere wenn kein Mensch in der Lage wäre, das Missverständnis aufzulösen. Man könnte argumentieren, dass diese Fehler auftreten, weil die künstliche Intelligenz sich noch in einer frühen Entwicklungsphase befindet, und dass es nur eine Frage der Zeit ist, bis diese Fehler nicht mehr auftreten – aber können wir uns darauf verlassen?

Komplexe Algorithmen sind nicht perfekt

1. Damit ein Satz in der Mathematik wahr ist, muss er bewiesen werden.
2. Aber was ist mit: „Dieser Satz lässt sich nicht beweisen“?

Wenn dieser Satz wahr ist, dann lässt er sich nicht beweisen und ist nach mathematischen Kategorien falsch.

Die schlechte Nachricht: Von Maschinen geschaffene komplexe Algorithmen sind ebenfalls Systeme und daher unvollständig, unvollkommen, fehlerhaft und enthalten unerwartete Bugs.

Die gute Nachricht: Ein Schachalgorithmus muss nicht alle möglichen Spiele gewinnen. Er soll lediglich öfter gewinnen als der aktuelle Weltmeister. Wenn Menschen ein Gesundheitssystem als Brettspiel entwerfen könnten, dann würden Maschinen immer bessere Möglichkeiten finden, dieses Spiel zu gewinnen, so wie AlphaGo Zero es schaffte, das Brettspiel Go zu gewinnen, besser, als jeder Mensch es könnte.

Was Entwickler und Architekten leisten müssen, ist die Darstellung von Systemen als Brettspiele – es stellt sich aber die Frage, wie das zu erfolgen hat. Die Risiken der Software sind unabhängig vom Entwicklungsansatz dieselben und müssen daher

einkalkuliert werden.

Testbasiertes Reinforcement Learning

Bei der traditionellen Entwicklung von Machine Learning wird ein Datensatz in Trainingsdaten (80 %) und Testdaten (20 %) aufgeteilt. Während die Trainingsdaten verwendet werden, um die Maschine zu trainieren, dienen die Testdaten dazu, zu bewerten, wie genau eine Maschine vorhersagt/kategorisiert. Daten, die die Genauigkeit verringern, werden aus dem Datensatz entfernt, während Daten, die die Genauigkeit steigern, hinzugefügt werden. Das Aufbauen, Navigieren und Verbessern der Datensätze erfordert einen Data Scientist oder jemanden mit umfassendem mathematischem, statistischem und technischem Know-how. Dies schließt viele Menschen von der Arbeit mit Machine Learning aus.

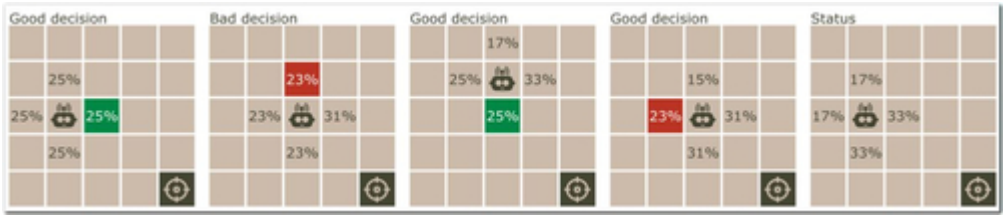

Abb. 2: Visualisiertes Beispiel, wie mein Prototyp lernen und verlernen kann

Beim Reinforcement Learning geht es darum, Verhaltensweisen zu belohnen oder zu bestrafen, um erwünschtes Verhalten zu bestärken und unerwünschtes Verhalten zu verringern. Ich habe einen Prototyp entwickelt, der dies realisiert: Ein Bot, der ein Ziel verfolgen muss. Der Bot verfügt über vier Kameras (eine in jede Richtung) und ein Bein (damit er sich in eine Richtung bewegen kann). Der Bot beginnt damit, das Bein nach dem Zufallsprinzip zu bewegen. Um eine Beinbewegung zu bestärken, könnte ein Entwickler einen technischen Test aufsetzen, der den Abstand zwischen dem Bot und seinem Ziel misst. Verringert sich die Distanz, wird die Wahrscheinlichkeit der Beinbewegung bestärkt, nimmt der Abstand hingegen zu, sinkt die Wahrscheinlichkeit der Beinbewegung (**siehe Abb. 2**):

Mein Prototyp kann aufgrund des kontinuierlichen Lernens und Verlernens sogar mit Änderungen in der Umgebung umgehen. Wenn zwei Kameras den Platz tauschen, verlernt der Bot die alten Kamerabewegungsbeziehungen und erlernt die neuen.

Ich nenne diesen Ansatz TDRL für „Test Driven Reinforcement Learning“. Er unterscheidet sich grundsätzlich von der traditionellen Entwicklung des Machine Learning. Anstatt den Datensatz für eine Maschine aufzubereiten, erhält die Maschine Zugang zu einem ungefilterten Datensatz oder einer Umgebung. Die Tests steuern die Maschinenentwicklung, und immer dann, wenn das Verhalten einer Maschine geändert oder korrigiert werden muss, kann ein neuer Testfall geschrieben werden. Beispielsweise setzte Microsoft einen KI-Chatbot namens „Tay“ ein, der innerhalb von 24 Stunden begann, rassistische Verhaltensweisen zu zeigen. Wäre dieser Chatbot mit TDRL aufgebaut worden, hätte man dies auf einfache Weise lösen können, indem man einen Testfall hinzugefügt hätte, der den Grad an Rassismus innerhalb eines Satzes misst. Der Chatbot hätte dann das Problem alleine gelöst, indem er seine eigenen Sätze geregelt und die Wahrscheinlichkeit verringert hätte, etwas mit rassistischem Inhalt zu schreiben.

Im Buch „Testing in the digital age“ [Tes2018] wird erläutert, dass wir nicht nur technische Tests brauchen (wie im obigen Beispiel „Distanz messen“), sondern auch ethische und konzeptionelle Tests (wie das Messen von Stimmungen, Empathie, Humor und Charme), um Roboter in der Rolle als Partner, Kollege oder Assistent zu verbessern. Die Entwicklung technischer Tests ist weniger komplex, während ethische und konzeptionelle Tests schwieriger zu entwickeln sein werden. Letztere würden auch domänenspezifisches Wissen erfordern. So lässt sich z. B. das Konzept „Gesundheit“ oder „Glück“ unterschiedlich wahrnehmen: Was für mich gesund ist, ist nicht notwendigerweise für meine Kinder oder die Umwelt und unseren Planeten gesund.

Zusammenfassung

Wie wird die künstliche Intelligenz die Softwareentwicklung beeinflussen und sich somit auf das Berufsbild der Entwickler und IT-Architekten auswirken?

Es besteht kein Zweifel, dass die künstliche Intelligenz die IT-Branche in den 2020er-Jahren tiefgreifend verändern wird.

Maschinen werden in der Lage sein, in kürzerer Zeit bessere und komplexere Lösungen zu entwickeln, als dies menschliche Softwareentwickler oder IT-Architekten können. Maschinen werden sogar die Welt der Data Scientists auf den Kopf stellen.

Was haben wir von der Welt der 2020er-Jahre zu erwarten?

Schon heute ist die Notwendigkeit sichtbar, Machine-Learning-Bots zu steuern. Und diese Notwendigkeit wird noch größer werden. Um die Maschinen, die Software entwickeln, zu kontrollieren, müssen IT-Entwickler und -Architekten Tests ausarbeiten und die Maschinen bei der Entwicklung von Software und Lösungen anleiten.

Wie sollten wir uns darauf vorbereiten?

Es ist schon heute notwendig, ethische Tests zu entwickeln, wie bspw. für den KI-Chatbot „Tay“ von Microsoft, der rassistisch wurde. Bevor aber ethische Tests durchgeführt werden können, muss aus den aktuellen TDLR-Prototypen eine neue Generation von Entwicklungstools entstehen. Als Entwickler oder Architekt hat man in den kommenden Jahren die Gelegenheit, Teil dieser Evolution zu werden und die Erfahrung zu sammeln, die man braucht, um nicht nur die technischen, sondern auch die ethischen und konzeptionellen Tests zu gestalten. Das Buch „Testing in the digital age“ [Tes2018] stellt einen guten Anfang dar, um herauszufinden, wie man diesen Herausforderungen am besten begegnet. Ethische und konzeptionelle Tests werden uns abverlangen, schwierige domänenspezifische Fragen zu beantworten, wie etwa: „Was ist ein guter Partner?“, „Was zeichnet ein gutes Produkt aus?“, „Wie macht man den Kunden glücklich?“ usw. Diese Fragen sind wichtig, und als Entwickler oder Architekten in den 2020er-Jahren werden wir sie beantworten!

Literatur & Links

[Cor57] "The Perceptron A perceiving and recognizing automaton (project para)", Cornell Aeronautical Laboratory, inc., Bericht Nr. 85-560-1, Frank Rosenblatt, Januar 1957

[Pat61] "A Pattern Recognition Program That Generates, Evaluates and Adjusts its Own Operators", Leonard Uhr and Charles Vossler, Mai 1961

[Che] http://mathematics.chessdom.com/shannon-number

[Uni] https://www.universetoday.com/36302/atoms-in-the-universe

[Ifl] http://www.iflscience.com/technology/artificial-intelligence-dreams/

[Pix] https://petapixel.com/2017/12/05/ai-can-change-weather-seasons-time-day-photos/

[Dep] https://deepmind.com/documents/119/agz_unformatted_nature.pdf

[Tes2018] Testing in the digital age, by Rik Marselis, Tom van de Ven, Humayun Shaukat, 2018, ISBN 9789075414875

Bartek Rohard Warszawski

definiert „Wissen“ als die Fähigkeit, eine Zukunft vorherzusagen – wer etwas prognostizieren kann, kann sich auch darauf vorbereiten. Er verfügt über nahezu 30 Jahre Erfahrung im Bereich der Softwareentwicklung und des Testing und greift auf diese Erfahrung zurück, wenn er auf Konferenzen spricht, Artikel schreibt und Workshops in testgetriebener Entwicklung (TDD) und Testautomatisierung durchführt.

E-Mail: Bartek.Warszawski(at)Capgemini.com

Bildnachweise:

Bartek Rohard Warszawski