Duluman Andrada-Georgiana
Grupa 331

RAPORT ARTICOL ȘTIINȚIFIC [1]

1. INTRODUCERE

Dezvoltarea Software conține diverse tipuri de erori, fie ca vorbim de faptul că cerințele pot fi înțelese greșit, de micile greșeli legate de cod ale developerilor sau de componentele software afectate, proiectele pot deveni nefuncționale. Așa a apărut ca reacție Testarea Software. Cu toate acestea, testarea eficientă a sistemelor software pentru a asigura calitatea produsului reprezintă o provocare, iar acest lucru se datorează faptului că este costisitoare, dar și consumatoare de timp. Astfel, a fost introdusă testarea automată.
Însă, pentru anumite aplicații, simpla testare automată nu a fost de ajuns. De aceea, aceasta a fost îmbunătățită introducându-se Machine Learning, atât supervizată, cât și nesupervizată, în cazuri de testare precum: fuzzing, mutation, generarea input-urilor și multe altele. Cu toate acestea, pentru scenariile complexe cu un număr infinit de stări, această soluție nu face față. De aceea, este introdusă învățarea prin recompensă (Reinforcement Learning, RL), iar articolul științific dat prezintă o analiză asupra acestei idei, unde este RL-ul folosit în testarea sistemelor software și care sunt avantajele și dezavantajele.

2. BACKGROUND

Pentru început, în articol se face o scurtă prezentare a celor 3 concepte: testarea software, machine learning și reinforcement learning.
Un test case reprezintă un set de condiții ce ar trebui să fie îndeplinite pentru a evalua corectitudinea și performanța unui sistem sau a unei aplicații software. Pentru că testarea manuală presupune executarea de către oameni și deci sursa principală a greșelilor, se preferă automatizarea acestui proces. Astfel, articolul prezintă tehnicile de testare ca fiind împărțite în 3 mari categorii: ”analysis testing” - testele folosite pentru a analiza codul, ”test execution” - tehnicile de executare eficientă a testelor (în mare parte testarea automată) și ”test generation” - tehnici de generare a input-urilor ce vor fi folosite în testare (ex. tehnica fuzzing).

În continuare este prezentat termenul de machine learning ca fiind o umbrelă pentru supervised, unsupervised, semi-supervised, și reinforcement learning.
Supervised Learning (SL) reprezintă un algoritm de machine learning ce necesită un set de date de antrenament care conține, atât caracteristicile de intrare, cât și valoarea de predicție. Astfel, o abordare de tip SL necesită intervenția umană pentru a eticheta setul de date, dar și pentru a selecta algoritmul și parametrii potriviți.
Semi-Supervised Learning (SSL) se bazează pe aceeași ideea ca SL, doar că având doar 10%-20% din setul de date de antrenament etichetat, iar restul neetichetat.

Unsupervised learning (UL), spre deosebire de SL, este un model matematic care utilizează seturi de date de antrenament ale vectorilor de intrare și nu include o variabilă țintă.
În articol se prezintă Deep Learning-ul (DL), recent introdus, ca fiind un subset al machine learning-ului și având la bază tehnici mai avansate precum conceptul de neuroni și Multi-Layer Perceptron ce au condus la o mai bună aproximare.
În urma prezentării, atât a SL, cât și UL, se aduce în vedere faptul că acestea nu sunt suficiente pentru testarea interactivă, de exemplu pentru testarea de GUI a aplicației. Astfel, autorii articolului afirmă că soluția la această problemă ar fi Reinforcement Learning-ul (RL) ce poate învăța într-un mediu interactiv.

Astfel, se prezintă RL-ul ca fiind tehnica de machine learning pentru un agent de a obține informații, însă fără să necesite un set de date de intrare/antrenament, ci direct din interacțiunea agentului cu mediul, fiind bazat pe Procesul de Decizie Markov (MDP). Astfel, agentul se poate afla într-una din multe stări ale mediului, poate alege una din mai multe acțiuni de a le face pentru a ajunge în altă stare, iar în cele din urmă primește o recompensă în urma acțiunii luate. Algoritmii de RL pot fi de 3 tipuri: *value-based* - se actualizează funcția de valoare a acțiunii pe fiecare pas în funcție de valoarea anterioară (Q-learning, DQN); *policy-based* - actualizează o politică la sfârșitul episodului de învățare (PPO); *actor-critic* - folosește puterea de învățare a celor două menționate anterior pentru a diminua pe cât posibil variația și bias-ul produs de ele.

3. RESEARCH ȘI REZULTATE

Autorii articolului prezintă cele 5 întrebări pe care le-au formulat pentru a răspunde la ele în continuare:
Q1. Care sunt aplicațiile RL în testarea software-ului?
Q2. Care sunt algoritmii și arhitecturile RL utilizate în mod obișnuit pentru învățare?
Q3. Care sunt provocările cu care se confruntă utilizarea RL în contextul testării software?
Q4. Care sunt avantajele și dezavantajele utilizării RL în testarea software-ului?
Q5. Care este performanța RL în comparație cu alte tehnici?
Aceștia prezintă procesul amănunțit de alegere, selecție și filtrare a setului de date, de lucrări scrise pe acest topic (utilizarea RL în testarea software) pentru a le analiza în detaliu și a ajunge să răspundă la cele 5 întrebări, dar și pentru a formula concluzii în acest sens.
Astfel, se prezintă răspunsul la care au ajuns aceștia pentru fiecare întrebare propusă.

Q1. Care sunt aplicațiile RL în testarea software-ului?

Duluman Andrada-Georgiana
Grupa 331

În urma analizei aplicate de autori, aceștia afirmă faptul că RL este cel mai des întâlnit în testarea interfețelor GUI, fiind urmată de generarea datelor de testare și selectarea cazurilor de testare pe locul doi, respectiv trei, iar mai apoi testele legate de robotică.

Q2. Care sunt algoritmii și arhitecturile RL utilizate în mod obișnuit pentru învățare?
Din rezultatele analizei realizate de autorii articolului reiese faptul că cel mai folosit algoritm de RL în testarea software este cel de Q-learning, care este un algoritm bazat pe valori. Acesta este urmat de algoritmi ce se bazează pe Deep Learning, precum: Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), Soft Actor–Critic (SAC) și alții. Cu atât mai mult, s-a observat faptul că majoritatea implementărilor de algoritmi RL sunt bazați pe un singur agent (single-agent).

Q3. Care sunt provocările cu care se confruntă utilizarea RL în contextul testării software?
O primă provocare prezentată este constituită de nevoia de informații de antrenament necesar algoritmului. Dacă informațiile nu sunt suficiente, în special pentru testarea black-box, poate împiedica capacitatea de a aplica în mod eficient RL-ul, iar dacă este prea multă informație poate duce la o creștere exponențială a timpului necesar pentru antrenare. O altă provocare discutată este incapacitatea de a aplica un model RL antrenat de la o aplicație la alta fără a fi reantrenată, fapt ce consumă timp de fiecare dată.

Q4. Care sunt avantajele și dezavantajele utilizării RL în testarea software-ului?
Principalele avantaje deduse sunt: automatizarea procesului de explorare pentru a găsi cele mai bune configurații (de input-uri) și minimizarea efortului dezvoltatorilor în crearea unor cazuri de testare complexe. Pe de altă parte, dezavantajele constau în: complexitatea proiectării mediului corect, ajustarea hiperparametrului pentru a ajunge la cel mai bun model, dar și timpul sau numărul de iterații pentru antrenament, în special în medii complexe.

Q5. Care este performanța RL în comparație cu alte tehnici?
În urma analizei autorilor reiese că în mai bine de 37% din cazuri abordările bazate pe RL au dat o acoperire mai bună în domeniile respective, iar în 22% timpul necesar pentru dezvoltarea modelului RL în comparație cu dezvoltarea manuală a cazurilor de testare a fost redus, obținându-se o acuratețe și eficiență mai mare în ceea ce privește execuția (11%).

4. DISCUȚII

Analizând folosirea algoritmilor de RL, autorii au ajuns la concluzia că aceștia sunt folosiți în principiu pentru două tipuri de testări: testarea GUI a aplicațiilor (fie web, desktop sau mobile) și generarea datelor de testare. De asemenea, RL mai este folosit și în cazul testării compiler-ului, testarea memoriei și testarea concurentă, însă nu la fel de mult.

Autorii articolului afirmă că folosirea RL-ului în testarea GUI vine cumva în mod natural având în vedere că testarea GUI se potrivește cu modul în care funcționează un algoritm RL (prezentat mai sus). În ceea ce privește generarea datelor de testare, RL a fost folosit pentru a crea inputuri ce ar acoperi cât mai multe sau ar crea excepții, fapt pe baza căruia sunt oferite și recompensele.

Ca o observație generală, aceștia spun că cel mai folosit algoritm de RL în testarea de sisteme software este cel de Q-learning, iar acest lucru se datorează faptului că acest algoritm este simplu și suficient pentru sisteme cu un număr mic de stări. Cu toate că în unele cazuri acesta aduce rezultate optime sau semi-optime, prezintă și provocări precum alegerea adecvată a reprezentării stării, costul antrenării agentului, dar și alegerea potrivită de hiperparametrii.

Un alt tip de testare în care se poate folosi RL este testarea exhaustivă, ce constă în implementarea unui set de teste ce acoperă întregul sistem întocmai pentru a testa funcționalitățile acestuia. Un avantaj al RL-ului în acest sens constă în capacitatea acestuia de a învăța proprietățile meta-euristice automat, ceea ce conduce la prioritizarea și generarea de teste optime. Însă pentru a testa un sistem e nevoie de o bună înțelegere a acestuia și a funcționalităților lui, deci o testarea black-box reprezintă o provocare. Cu toate acestea, datorită naturii exploratoare a RL-ului, dar și a abilității de a învăța, acesta reprezintă un avantaj și o soluție pentru testarea de acest tip.

Pe de altă parte, avem schimbările ce pot apărea în testarea automată GUI, testarea de regresie, dar și sisteme cu mai mulți useri ale căror acțiuni concurente pot conduce la o cădere a acestora. Astfel, este nevoie de o generare de teste care să ia în calcul toate posibilitățile, iar avantajul RL-ului este că reduce timpul pentru dezvoltare, dar și capacitatea acestuia de a se extinde și de a crește automat odată cu creșterea sistemelor și a funcționalităților acestora.

Mai sunt câteva cazuri de testare non-tradițională în care este luată în considerare folosirea RL-ului, și anume testarea vehiculelor autonome pentru siguranță și testarea de regresie în jocuri. Autorii afirmă faptul că acestea sunt încă în dezvoltare și se îndreaptă într-o direcție bună, însă momentan potențialul RL-ului este evidențiat în testarea GUI.

Aplicabilitatea în testarea software a acestor algoritmi de RL se regăsește și în adaptarea cazurilor de testare, programarea lor, testarea de regresie și integrarea și implementarea continue. De asemenea, așa cum au mai precizat în lucrare autorii, algoritmii de RL au aplicabilitate în testarea tip white-box pentru optimizarea selecției cazurilor de testare pe baza impactului așteptat asupra calității software-ului de

exemplu, însă și în testarea black-box cu cât abordările RL sunt mai potrivite pentru sarcinile care implică luarea deciziilor în condiții de incertitudine.

Un alt aspect luat în considerare constă în spațiile mai mari de acțiune în care o abordare de tip deep-RL este mai potrivită decât una tradițională. Astfel, autorii afirmă că în acest caz este mai bine să existe mai mulți agenți RL care interacționează într-un mediu dinamic. Așadar, aceștia afirmă faptul că alegerea abordării RL este legată în principiu de constrângerile și cerințele sarcinii.

De asemenea, pentru lucrări viitoare, autorii propun explorarea altor algoritmi de RL pentru testarea software decât cel de Q-learning pentru a găsi soluții mai eficiente, analiza acestora pentru găsirea configurațiilor potrivite, dar și discuții mai ample legate de folosirea acestor algoritmi în testarea vehiculelor sau a roboticii.

5. CONCLUZII

În concluzie, se poate afirma faptul că testarea software-ului a devenit un element crucial în asigurarea dezvoltării software de înaltă calitate, însă pentru unele scenarii complexe, nici metodele de SL, nici UL nu au fost adecvate, de aceea o soluție este reinforcement learning (RL). Astfel s-a arătat faptul că RL a fost utilizat pe scară largă în testarea software-ului, dar aproape că s-a restrâns la două aplicații, au fost evidențiate provocările aplicării algoritmilor de tip RL, dar și avantajele acestora în comparație cu cele de bază din testarea software.

REFERINȚE

[1] Amr Abo-eleneen, Ahammed Palliyali, Cagatay Catal, *The role of Reinforcement Learning in software testing,* 2023, (14 pagini)