Studiul și prezentarea unui articol științific

Silviu-Andrei Matu
Anul III Informatică ID
Testarea Sistemelor Software
Universitatea din București, Facultatea de Matematică și Informatică
Email: silviu-andrei.matu@s.unibuc.ro

14 iunie 2025

Cuprins

1 Introducere

Articolul pe care l-am analizat pentru acest proiect este intitulat „Evaluating and Improving ChatGPT for Unit Test Generation" [3] și a fost redactat de o echipă de cerecetători afiliați Uiversității Fundan din China. Articolul discută motivația pentru utilizarea *Large Language Models (LLMs)* pentru generarea de *Unit Tests (UTs)*, prezintă performanța actuală a unui dintre cele mai populare modele LLM, anume *ChatGPT*, după care evaluează un proces iterativ de generare a UTs care poate valorifica orice model LLM capabil să genereze cod în limbajul de programare vizat. În articolul prezent, autorii s-au concentrat pe generarea de teste în *Java*, cu o evaluare secundară realizată în *Python*.

2 Motivație pentru testarea automată folosind LLMs

Argumentele la care autorii fac apel pentru a justifica de ce merită investigate LLMs în ceea ce privește capacitatea lor ge a genera UTs vizează desigur ideea automatizării testării și a reducerilor de costuri pe care aceasta le-ar putea aduce. Dar autorii punctează și faptul că sistemele de automatizare existente, cum sunt de exemplu cele de tip *seach-based* și soluțiile ce folosesc *deep learning* tind să genereze teste care nu sunt interpretabile ușor de către programatorii umani și ca urmare sunt rareori adopte în practică.

3 Testarea soluțiilor curente (Studiul 1)

Scopul final al articolului este de a propune o nouă abordare pentru dezvoltarea de UTs, dar pentru a susține eficiența acesteia, autorii propun mai întâi stabilirea unui *benchmark* la care să se raporteze. Acest reper a fost reprezentat de utilizarea unui model LLM comercial foarte popular, *ChatGPT* pentru a genera UTs, pornind de la un prompt simplu care indică contextul („Ești un programator ...", textul metodei țintă și antetul clasei din care face parte, alături de antetele celorlalte metode din clasă). Pentru e cuantifica capacitatea acestei abordări de a genera teste utile în sarcini de programare, au cuantificat performanța pe patru dimensiuni:

- cât de corecte sunt testele generate. Această dimensiune a fost reflectată de numărul de teste fără erori de sintaxă și numărul de teste care au trecut de etapa de compilare.
- cât de bine acoperă codul. Această dimensiune s-a reflectat în acoperirea ramurilor din cod de către testele generate.
- cât de interpretabile sunt testele generate. Această dimensiune a fost evaluată de o serie de evaluatori umani.

– cât de utilizabile sunt testele generate. Din nou, această dimensiune a fost evaluată de către utilizatori umani care au raportat experiența lor folosind un chestionar standardizat.

În ceea ce privește baza de date pe care au folosit-o pentru a realiza testarea, aceasta a fost construită pentru studiul de față pornind de la o selecție de proiecte publice cu evaluări pozitive care, pe lângă codul propriu-zis al programelor, ofereau și testele asociate metodelor pe care le conțineau. Cu această abordare autorii au construit o bază de date cu N = 1000 de metode și teste asociate. Pentru fiecare dintre aceste metode au fost generate teste alternative cu ajutorul ChatGPT, folosind instrucțiunile menționate mai sus. Pentru a oferi un context mai larg de interpretare, autorii au mai folosit alte două instrumente de generare automată de teste. Una dintre ele poartă denumirea de AthenaTest, fiind o metodă care are la bază arhitectura de tip *transformer* ca și majoritatea modelelor LLM [2]. Cea de a doua este o metodă de căutare a soluțiilor folosind algoritmi genetici, numită Evosuite [1]. Aceste abordări au fost utilizate în studii anterioare și reprezintă punct de reper relevante pentru a evalua capacitatea ChatGPT.

Rezultatele au arătat că ChatGTP a generat teste în totalitate corecte din punct de vedere sintactic, dar majoritatea nu a trecut de compilare și doar o mică parte au fost executate (24.8%). AthenaTest a avut o performanță mai slabă la toți indicatorii (ex. doar 14.4% din teste au fost executate). Evosuite în schimb a avut un procent de 91.4% în ceea ce privește numărul de teste executate. Trebuie precizat că acest instrument din urmă, prin natura de funcționare, presupune un proces iterativ care generează și validează testele până ajunge la varianta finală.

În ceea ce privește acoperirea instrucțiunilor și a ramurilor din cod, testele generate de Chat GPT au acoperit 82.3% din instrucțiuni, respectiv 65.6% din ramuri. Procentele au fost de 65.5% și 56.2% pentru AthenaTest, respectiv 81.1% și 77.3% pentru Evosuite. Performanța testelor dezvoltate de utilizatorii umani colectate o dată cu metodele din baza de date au avut acoperiri medii de 84.2% și 68.9%. Autorii indică din nou faptul că performanța aparent mai bună a Evosuite cel puțin pe una dintre dimensiuni se datorează procesului iterativ de generare.

În ceea ce privește interpretabilitatea și utilizabilitatea testelor generate, autorii au cerut de la cinci evaluatori umani să evalueze comparativ testele generate manual și cele generate de ChatGPT, fără a ști cum a fost generat fiecare test. Rezultatele prezentate grafic au indicat faptul că cele două tipuri de teste au fost evaluate foarte similar de către toți cei cinci evaluatori.

4 Dezvoltarea și testarea ChatTester (Studiul 2)

Pornind de la rezultatele încurajatoare din studiul anterior, autorii au dezvoltate și au testat un proces iterativ de generare și rulare a unor UTs care au la bază ChatGPT sau

alte modele te tip LLMs. Ideea de bază a acestuia este de a folosi prompt-uri țintite și succesive pentru a depăși două probleme principale care au fost vizibile în testele din studiul anterior. În primul rând este necesară o înțelegere mai profundă a scopului pe care o are metoda pentru care se generează testul. În acest sens, autorii au folosit în primă fază un prompt de inițializare, care solicită modelului să identifice intenția din spatele metodei. Răspunsul generat este apoi folosit într-un nou prompt generativ care vizează construirea unei prime versiuni a testului. Acesta este analizat sintactic, compilat și rulat. Dacă apar probleme în compilare sau rulare, atunci eroare este extrasă și folosită ca și prompt pentru a genera o variantă îmbunătățită a testului, în speranța că aceasta va trece de barierele anterioare. Pasul de generare a unei versiuni îmbunătățite dacă apare o eroare este repetat fie până când testul este rulat complet, sau până când se atinge un număr de iterații predefinit. Întregul proces iterativ de generare a fost denumit *ChatTester*.

Pentru a testa noua abordare, autorii au construit un al doilea set de metode extrase din arhive software publice, de data aceasta însă mai restrâns, N = 185. Pe lângă metoda completă, cu toate tipurile de prompt-uri, au fost include în teste și o serie de variante restrânse, în care a fost eliminată fie componenta de identificare a intenției, fie componenta iterativă de corectare a erorilor. De asemenea, pentru a testa dacă procesul de generare este dependent de modelul LLM pe care se bazează, autorii au construit variate alternative acare au folosit alte modele populare pentru generarea de cod, anume *CodeLLama* și *CodeFuse*. Metodologia de testare a fost similară în ceea ce privește parametrii vizați: corectitudinea testelor, gradul lor de acoperire, interpretabilitatea și utilizabilitatea. Ultimele două componente au fost evaluate cu aceiași metodă de tip chestionar ce a fost completat de cinci programatori cu experiență.

Rezultatele au arătat că performanța ChatTester care are la bază modelul ChatTGPT a fost dublă față de modelul ChatGPT simplu în ceea ce privește numărul de teste care au fost compilate corect (de la 39.0 la 73.3%) și acelor care să fie executate (de la 22.3 la 41.0%). Variantele intermediare, care nu au folosit procesul iterativ de corectare al erorilor sau care nu au folosit intenția ca și prompt au avut performanțe mai slabe; doar 29.7% și 34.0% din teste au fost executate cu succes. Analiza numărului de teste care au trecut de compilare sau de execuție după un număr mai mare de iterații a sugerat că există teste care sunt îmbunătățite și trec de aceste etape după un număr mai mare de corecții. Rezultatele modelor care nu folosesc toate tipurile de prompt-uri și evoluția testelor în funcție de numărul iterații converg spre idea că toate componentele ChatTester contribuie la performanța acestuia. Performanța s-a îmbunătățit și atunci când au fost comparate testele generate de celelalte modele LLM cu performanța celor generate de aceleași modele dar folosind procedura de generare a prompt-urilor presupusă de ChatTester. În ceea ce privește nivelul de acoperire al testelor, autorii folosesc din nou ca puncte de reper aceleași modele testate în Studiul 1. În aceste teste, realizate pe noua bază de date, ChatTester

a avut performanțe similare celor ale Evosuite. În fine, în ceea ce privește nivelul de utilizabilitatea și interpretabilitate, autorii indică faptul că performanța ChatTester a fost peste cea a celorlalte modele cu care a fost comparat. ChatTester a arătat îmbunătățiri similare și în cazul testelor redactate în limbajul de programare Python.

5 Discuții și concluzii cu privire la generarea de teste folosind LLMs

Articolul prezintă o investigare extensivă a potențialului utilizării ChatGPT și a altor modele de tip LLM pentru testarea automată. Ideile centrale pe care le scot în evidență rezultatele prezentate sunt acelea că aceste modele pot genera teste care sunt utilizabile și interpretabile, ceea ce crește șansa ca ele să fie adoptate în practică, respectiv faptul că aceleași modele pot fi folosite iterativ pentru a genera prompt-uri mai bune, care să le crească performanța. Trebuie menționat că modelele utilizate nu au fost antrenate specific pentru această sarcină, ceea ce sugerează că acestea ar putea să fie îmbunătățită mai departe prin procese de *fine tuning*. Într-o notă mai critică, menționăm că unele rezultate nu sunt prezentate foarte detaliat pentru a judecat natura diferențelor care există între modele. E posibil ca unele dintre rezultatele prezentate ca fiind pozitive să fie variații care pot fi atribuite șansei.

Bibliografie

[1] Gordon, F. and Andrea, A. (2011). EvoSuite | Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering.

[2] Tufano, M., Drain, D., Svyatkovskiy, A., Deng, S. K., and Sundaresan, N. (2020). Unit Test Case Generation with Transformers and Focal Context.

[3] Yuan, Z., Liu, M., Ding, S., Wang, K., Chen, Y., Peng, X., and Lou, Y. (2024). Evaluating and Improving ChatGPT for Unit Test Generation. *Proceedings of the ACM on Software Engineering*, 1(FSE):1703–1726.