GENERAREA DE COD CU AJUTORUL INTELIGENȚEI ARTIFICIALE: RISCURI ȘI BENEFICII

Artiom COMANAC

Ingineria Software și Automatică, SI-231M, Facultatea Calculatoare, Informatică și Microelectronică, Universitatea Tehnică a Moldovei, Chișinău, Republica Moldova

Autorul corespondent: Comanac Artiom, comanac.artiom@isa.utm.md

Îndrumătorul/coordonatorul științific **Rodica BULAI**, lector universitar

Rezumat. *Articolul analizează modelele de inteligență artificială pentru generarea codului, devenite extrem de populare și actuale. Aceste instrumente promit să revoluționeze dezvoltarea software-ului și să înlocuiască un număr mare de programatori, doar ca utilizarea lor este asociată cu un număr mare de riscuri de securitate și necesită să fie minuțios analizată. În lucrare sunt prezentate cele mai populare instrumente de generare a codului, cum ar fi GitHub Copilot, CodiumAI, Amazon CodeWhisperer și cel mai recent apărut, Devin. Cercetarea este axată pe generarea securizată a codului și asigurarea confidențialității lui. Sunt evidențiate avantajele și dezavantajele aplicării inteligenței artificiale în dezvoltarea software-ului, inclusiv utilizarea lor în companii mari și zone critice. De asemenea, sunt scoase în evidență posibilele erori care apar la generarea automată a codului, vulnerabilitățile sau scurgerile de date sensibile datorită modelelor de învățare automată.*

Cuvinte cheie: *inteligența artificială, instrumente de generare automată a codului, copilot, devin, codium.*

Inteligența artificială si generarea codului

În ultimii ani, inteligența artificială și prelucrarea limbajului natural au înregistrat progrese semnificative [1]. Acest lucru a condus la dezvoltarea rapidă și activă a inteligenței artificiale generative, pe baza căreia au fost construite majoritatea instrumentelor pentru generarea de cod. În prezent, există deja o mulțime de astfel de asistenți, cum ar fi GitHub Copilot [2], CodiumAI [3], Amazon CodeWhisperer și recentul Devin, al cărui creatori îl numesc primul dezvoltator autonom de cod [4]. În funcție de necesitați, capacități și resurse, dezvoltatorii și companiile aleg diverse instrumente, majoritatea dintre acestea permit lucrul în mediul „enterprise” [5]. Popularitatea și utilizarea acestor instrumente crește vizibil conform cercetărilor pe piață privind IA generativă, figura 1.

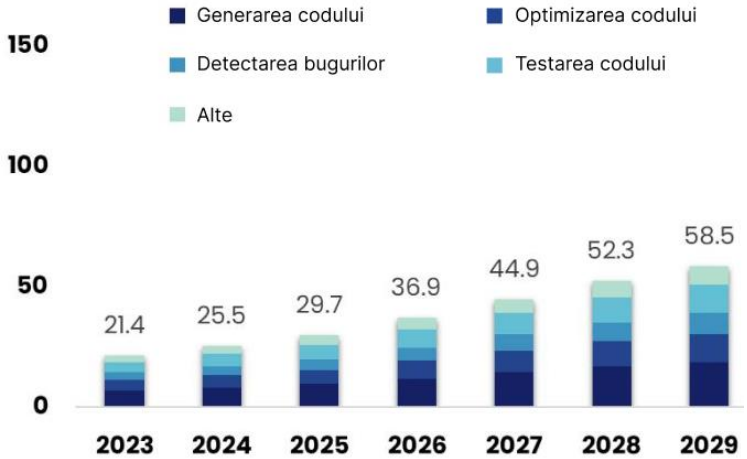

Figura 1. Cercetarea utilizării inteligentei artificiale în dezvoltarea softului [6]

Acești asistenți promit să revoluționeze lumea programării și dezvoltării de software, oferind diverse îmbunătățiri și asistență dezvoltatorilor, dar în același timp ascund diverse riscuri și pericole, care trebuie luate în considerare în timpul lucrului cu ei.

Generarea vulnerabilităților si erori in cod

Utilizarea instrumentelor de generare a codului bazate pe inteligență artificială este asociată cu diverse riscuri, ceea ce impune anumite restricții în utilizarea acestor instrumente. Unul dintre riscurile asociate cu utilizarea instrumentelor bazate pe inteligență artificială este generarea erorilor în cod. Modelele de generare a codului au fost antrenate pe un set de cod mare, însă nu întreg setul de date pentru învățare reprezintă un exemplu de bune practici (best practice), iar în același timp este dificil să se determine ce reprezintă o practică bună, deoarece acestea se modifica și se completează foarte des. De exemplu, pentru învățarea GitHub Copilot au fost utilizate toate repozitoriu-rile deschise din GitHub, și nu există nicio garanție că a fost utilizat doar cod adecvat [7]. În același timp, pentru generarea codului, instrumentele au nevoie de un context, adică o parte a codului pentru care trebuie să genereze sau să modifice codul, ceea ce înseamnă că aceste instrumente pot completa codul eronat scris de utilizator. Acest lucru duce la faptul că instrumentele generează cod eronat pe baza a ceea ce are utilizatorul, ceea ce implică riscul creșterii codului cu erori și a timpului de dezvoltare, deoarece este necesar timp suplimentar pentru identificarea și corectarea erorilor.

De asemenea, există riscul că instrumentele vor genera și diferite vulnerabilități în cod. Cel mai des acest lucru este specific codului în care există deja vulnerabilități. Instrumentele de generare utilizează codul cu vulnerabilități ca cod de bază și creează noi vulnerabilități în cod. De exemplu, dacă în proiect sunt folosite interogări către baza de date în mod direct, fără a folosi „placeholder”, acest cod este vulnerabil și există riscul destul de înalt că dacă se va folosi GitHub Copilot pentru a scrie cod pentru o nouă interogare, acesta ar putea lua ca bază codul existent cu vulnerabilități, iar acest lucru duce la crearea unei noi vulnerabilități în cod [8].

Un exemplu al unei astfel de generări poate fi observat în următorul exemplu:

```
String query = "SELECT * FROM items WHERE LOWER(name) like '%" + name
```

Cod existent cu posibile vulnerabilități

```
// create query to match item with description

String query = "SELECT * FROM items WHERE LOWER(description) like '%"
```

Prompt si cod generat de IA

Uneori instrumentele de generare nu recunosc vulnerabilități în codul existent și generează cod cu vulnerabilități noi. Acest lucru indică faptul că pentru a utiliza asistenții în scrierea codului este întotdeauna necesar verificarea rezultatul oferit de instrument.

Scurgerile de date

La baza fiecărui instrument și asistent în generarea codului se află o cantitate mare de date, pe baza cărora rețelele neuronale generează date noi. Pentru învățare, sunt folosite adesea date publice, cum ar fi repozitoriu-urile deschise, răspunsurile pe forumuri și altele. Foarte des, în timpul auto-învățării instrumentelor generative, nu sunt luate în considerare drepturile de utilizare a codului deschis, așa cum s-a descoperit, de exemplu, că GitHub Copilot a fost antrenat pe repozitoriu-urile deschise fără a ține cont de licențele lor. Acest lucru poate duce la faptul că în anumite cazuri, instrumentele pe baza inteligenței artificiale pot furniza o copie completă a codului sau chiar a comentariilor din cod. Acest lucru pune în pericol securitatea proprietății

intelectuale a programatorilor și a companiilor, deoarece nu există nicio garanție că codul privat nu va fi folosit pentru antrenarea rețelelor neuronale.

Cu toate acestea, conform asigurărilor creatorilor, citarea completă a codului este foarte rară și apare în doar 0.1% din cazuri [9]. Conform informațiilor din documentația Copilot, în cazul versiunilor Business și Enterprise, instrumentul nu folosește codul și datele introduse pentru învățare, dar în celelalte cazuri aceste date sunt utilizate pentru crearea setului de date pentru auto-învățarea rețelelor neuronale [10]. În același timp, instrumentele promit filtrarea datelor, cheilor API și altor chei de acces și fără furnizarea acestora în niciun fel.

Beneficii

Implementarea generatorilor de cod în dezvoltarea codului, în ciuda riscurilor și restricțiilor, poate ajuta specialistul să-și accelereze munca cu sarcini rutine, permițând să își dedice mai mult timp altor sarcini mai importante și mai consumatoare de resurse.

De exemplu, folosind generatorii de cod și diverse chat-uri Copilot, se poate accelera munca cu noi biblioteci și SDK-uri, precum și generarea rapidă a exemplelor de cod pentru lucrul cu diferite funcționalități. Acest lucru poate contribui la accelerarea lucrului cu codul nou și la integrarea mai rapidă a noilor funcționalități în cod [11].

Cu ajutorul instrumentelor bazate pe inteligență artificială, se poate efectua o analiză a codului pe baza principiilor și regulilor companiei, cum ar fi efectuarea unei revizuiri preliminare a codului [12]. Acest lucru poate ajuta la evitarea unui număr mare de erori simple și poate îmbunătăți stilul codului. De asemenea, acest lucru poate contribui la reducerea încărcării atunci când se verifică codul de către oameni reali, permițându-le să se concentreze asupra logicii mai complexe.

Testarea codului necesită adesea o cantitate mare de timp și poate fi automatizată cu ajutorul generatorilor de cod bazate pe inteligență artificială. Generatorii de cod pot crea teste foarte rapid și destul de precise pentru diferite funcții și clase în cod [13].

Un alt avantaj și posibilă aplicare a inteligenței artificiale este utilizarea sa pentru documentarea codului și scrierea comentariilor pentru funcții și clase. Economisirea timpului va permite concentrarea mai mare asupra scrierii codului, în loc să se documenteze acesta.

Concluzii

Integrarea în procesul de dezvoltare a instrumentelor pentru generarea codului bazate pe inteligență artificială aduce atât numeroase avantaje, cât și multe riscuri, majoritatea, în prezent, fiind dificil de evaluat. Pe de o parte, utilizarea acestor instrumente permite reducerea timpului de dezvoltare, concentrarea pe elemente mai importante și critice, familiarizarea mai rapidă cu modulele și bibliotecile noi. Ele facilitează analiza codului și găsirea soluțiilor mai optimizate.

Pe de altă parte, este necesar ca utilizatorii sa fie extrem de atenți și pregătiți pentru diverse erori în cod, erori de generare, precum și pentru posibile vulnerabilități în cod. Este esențială verificarea codul propus de aceste instrumente.

Utilizarea responsabilă a instrumentelor bazate pe inteligență artificială poate îmbunătăți productivitatea, reduce sarcina de muncă și optimizează colaborarea în echipă, dar în prezent nu e posibil bazarea completă pe dezvoltarea codului de către IA și întotdeauna este necesar ca codul generat să fie verificat cu atenție și să fie analizate toate riscurile asociate utilizării acestor instrumente.

Referințe

[1] O. Asare, M. Nagappan, și N. Asokan, „Is GitHub’s Copilot as bad as humans at introducing vulnerabilities in code?”, *Empir Software Eng*, vol. 28, nr. 6, p. 129, sep. 2023, doi: 10.1007/s10664-023-10380-1.

[2] Z. C. Ani, Z. A. Hamid, și N. N. Zhamri, „The Recent Trends of Research on GitHub Copilot: A Systematic Review”, în *Computing and Informatics*, N. H. Zakaria, N. S.

Mansor, H. Husni, și F. Mohammed, Ed., Singapore: Springer Nature, 2024, pp. 355–366. doi: 10.1007/978-981-99-9589-9_27.
[3] „Meaningful Code Tests for Busy Devs | CodiumAI”. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://www.codium.ai/
[4] „Devin AI Website - The First AI Software Engineer Cognition”. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://devinai.ai/
[5] „Your AI-powered chat for the Web with commercial data protection | Microsoft Copilot”. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://www.microsoft.com/en-us/bing/chat/enterprise/
[6] „Generative AI in Software Development Market Size & Share 2024”, MarketResearch.biz. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://marketresearch.biz/report/generative-ai-in-software-development-market/
[7] „GitHub Copilot Trust Center - GitHub Resources”. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://resources.github.com/copilot-trust-center/
[8] „Copilot amplifies insecure codebases by replicating vulnerabilities in your projects | Snyk”. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://snyk.io/blog/copilot-amplifies-insecure-codebases-by-replicating-vulnerabilities/
[9] A. Ziegler, „GitHub Copilot research recitation”, The GitHub Blog. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://github.blog/2021-06-30-github-copilot-research-recitation/
[10] „GitHub Copilot replicating vulnerabilities, insecure code | TechTarget”, Security. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://www.techtarget.com/searchsecurity/news/366571117/GitHub-Copilot-replicating-vulnerabilities-insecure-code
[11] „Optimising the Software Development Process with Artificial Intelligence | SpringerLink”. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://link.springer.com/book/10.1007/978-981-19-9948-2
[12] syedzainnasir, „Generative AI in Software Development: Boosting IT Productivity - The Engineering Projects”. Data accesării: 23 martie 2024. [Online]. Disponibil la: https://www.theengineeringprojects.com/2023/10/generative-ai-in-software-development-boosting-it-productivity.html
[13] A. Graaff, D. Smit, și S. Eybers, „Let’s Play Games: Using No-Code AI to Reduce Human Cognitive Load During AI Solution Development”, în *Artificial Intelligence Research*, A. Pillay, E. Jembere, și A. J. Gerber, Ed., Cham: Springer Nature Switzerland, 2023, pp. 86–99. doi: 10.1007/978-3-031-49002-6_7.