DOI: https://doi.org/10.36059/978-966-397-477-4-140

Олексій МАЦІЄВСЬКИЙ
асистент кафедри інформаційних технологій
Київського національного університету будівництва і архітектури (Київ)
matsiievskyi_oo@knuba.edu.ua

Тетяна ГОНЧАРЕНКО
доктор технічних наук з інформаційних систем та технологій, доцент,
завідувач кафедри інформаційних технологій
Київського національного університету будівництва і архітектури (Київ)
goncharenko.ta@knuba.edu.ua

ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ КОДУ: ІННОВАЦІЙНІ ПІДХОДИ ДО ОЦІНЮВАННЯ СТУДЕНТСЬКИХ РОБІТ

Активна інтеграція штучного інтелекту (ШІ) в освіту є ознакою наступного етапу розвитку інформаційних технологій. Автоматизація перевірки та тестування програмного коду, написаного студентами під час виконання лабораторних та практичних завдань, є із перспективних напрямків використання штучного інтелекту. Дослідження M. S. Kang, S. Y. Park, M.-A. Chung та D.-h. Han [1] показали, що досить ефективними є адаптивні алгоритми машинного навчання для пошуку помилок і аналізу продуктивності програм. Інструменти на основі штучного інтелекту, такі як AutoGrader, CodeGrade, Codio та Moss не лише перевіряють код відповідно до технічних вимог, але й оптимізують його, навчаючи студентів найкращим практикам програмування.

R. Acuña, T. Baron та S. Bansal аналізували AutoGrader, як інструмент, який виконуючи тестові сценарії, може автоматично оцінювати код, знаходячи логічні помилки та надаючи поради щодо покращення. Дослідження було проведено студентами на курсах розробки програмного забезпечення; результати показали, що автоматизовані інструменти оцінювання можуть бути корисними для поглиблених курсів комп'ютерної розробки та розробки програмного забезпечення [2].

Під час навчального процесу використання CodeGrade допоможе зробити процес перевірки простим і інтерактивним, а Codio дозволить студентам відразу бачити свої зміни в коді. Цей інструмент пропонує комплексні середовища для розробки та оцінювання коду. Moss корисний для виявлення плагіату, оскільки він допомагає викладачам знайти збіги у коді.

Цей підхід відкриває нові горизонти для індивідуалізації навчального процесу, оскільки дозволяє враховувати потреби кожного студента. Інтеграція штучного інтелекту у перевірку коду дозволяє кожному студенту отримувати не лише оцінку своєї роботи, але й індивідуальні рекомендації, які враховують його рівень знань, типові помилки та стиль програмування. Наприклад, якщо студент систематично допускає певний тип помилок, система не може лише вказати на них, але й надати покрокові поради для того, як їх можна уникнути в майбутньому.

У результаті викладачі не повинні виконувати рутинні завдання, такі як перевірка технічної відповідності та синтаксичних помилок. Це дозволяє їм приділяти більше уваги поясненню складних концепцій, створенню інтерактивних стратегій навчання та допомозі учням, які потребують додаткової підтримки. Машинний аналіз студентських робіт може використовуватися як засіб перевірки, так і як засіб діагностики загального рівня підготовки студентів, визначаючи теми, які потребують додаткового опрацювання.

Для викладача застосування машинного навчання є допомога визначити, які теми є актуальні для повторення. Алгоритми здатні виявити поширені шаблони помилок, такі як некоректне використання рекурсії або логічні помилки у структурах циклів, шляхом аналізу великої кількості студентських робіт. Це дозволяє не лише визначати помилки, але й пропонувати більш ефективні методи їх вирішення, використовуючи найкращі практики програмування, перевірені на практиці.

Статичний аналіз коду, який пропонують інструменти SonarQube або PyLint забезпечує перевірку його стилю написання: синтаксису та структури. Наприклад, система може перевірити, чи оптимізовані алгоритми коду або чи він відповідає стандартам стилю, таким як PEP8 для Python. З іншого боку, динамічний аналіз, реалізований через платформи Junit для мови програмування JAVA або pytest для Python, дозволяє тестувати програму на практиці, перевіряючи її поведінку на різних тестових даних.

Технології обробки природної мови, що використовуються в системах GPT-4 або Codex роблять написання коду користувачів більш зручними, розуміючи текстові коментарі, які студенти додають до свого коду. Наприклад, якщо студент коментує функцію, описуючи її призначення, система може перевірити, чи відповідає її фактична реалізація заявленій меті. Такі технології також можуть автоматично створювати пояснення помилок, що робить навчання більш простим.

Упровадження штучного інтелекту в автоматизоване тестування коду [3] відкриває нові можливості для розвитку освітнього процесу, роблячи його більш ефективним, гнучким і адаптивним. Розширення функціональності наявних інструментів і створення нових платформ, спрямованих на конкретні освітні потреби, є двома ключовими компонентами цієї інтеграції.

Сучасні системи можуть виконувати основні завдання, такі як статичний і динамічний аналіз коду, але вони також можуть додати студентам інтерактивне навчання, гейміфікуючи процес перевірки. Наприклад, платформи, такі як CodeCombat і CheckiO, підтримують процес написання коду в ігрових сценаріях, мотивуючи студентів до покращення своїх навичок.

Також слід звернути увагу на системи, які підтримують командну роботу, як-от GitHub Classroom. Завдяки цій платформі можна створювати навчальні репозиторії, автоматизувати перевірку завдань і надавати зворотний зв'язок. Це особливо важливо, коли студенти працюють над проектами, які нагадують реальні ситуації командної розробки програмного забезпечення.

Упровадження інструментів штучного інтелекту в процес оцінювання знань є ще одним напрямком розвитку. Алгоритми на основі глибинного навчання дозволяють створювати адаптивні тестові завдання, які автоматично адаптуються до рівня навчання студента. Це може забезпечити більш точну оцінку його знань, щоб визначити, де йому потрібно звернути більше уваги на свої знання.

Перспективними є також розвиток засобів для аналізу стилю та якості коду. Інструменти, які використовують алгоритми штучного інтелекту для передбачення та підказок у процесі програмування, такі як DeepCode або TabNine, можуть бути інтегровані в навчальний процес, щоб створити культуру написання якісного, зрозумілого та підтримуваного коду.

У той же час важливо враховувати моральні питання, пов'язані з використанням штучного інтелекту в освіті. Забезпечення прозорості алгоритмів оцінювання та захисту даних студентів мають бути першочерговими. Запровадження політик відкритості та регулярний аудит систем допоможуть уникнути упередженості алгоритмів і гарантувати їх справедливість.

Інструменти на основі штучного інтелекту в майбутньому повинні не лише оцінювати код, але й активно брати участь у навчальному процесі. Наприклад, інтерактивні асистенти на основі технологій, таких як GPT, можуть пояснювати складні поняття, надавати індивідуальні поради та навіть створювати навчальні матеріали, адаптовані до кожного студента.

Таким чином, автоматизація тестування коду за допомогою штучного інтелекту підвищує якість навчання, а також створює нові можливості для навчання майбутніх фахівців у сфері програмування. Такі технології є важливим кроком у персоналізації освіти, щоб протистояти викликам сучасного світу.

Література:

1. Machine Learning / M. S. Kang et al. *No-Code AI*. 2024. P. 15–41. URL: https://doi.org/10.1142/9789811293894_0002
2. Acuña R., Baron T., Bansal S. Autograder Impact on Software Design Outcomes. *2023 IEEE Frontiers in Education Conference (FIE)*, College Station, TX, USA, 18–21 October 2023. 2023. URL: https://doi.org/10.1109/fie58773.2023.10343266
3. Шкурко В., Поляков А. ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. *Радіоелектроніка та молодь у XXI столітті. Т. 7 : Конференція «Комп'ютерний зір, системний аналіз та математичне моделювання»*. Харків, Україна, 2024. URL: https://doi.org/10.30837/iyf.cvsamm.2024.311