3. Бойко І.В., Петрик М.Р., Цуприк Г.Б. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.

4. Methods of constructing algorithms for comparative test statistical verification of mathematical models of bioobject responses to low-intensity stimuli / Bohdan Yavorskyy, Evhenia Yavorska, Halyna Tsupryk, Roman Kinash // Scientific Journal of TNTU. — Tern.: TNTU, 2023. — Vol 112. — No 4. — P. 82–90.

5. Information System for Design of Thin Multilayer Film Processes Parameters Management based on Diffusion M.R., Petryk, Mykhaylo R., V., Chyzh, Vitalii, H.B., Tsupryk, H. B., O.Y., Petryk, Oksana Yu ITTAP'2024: 4th International Workshop on Information Technologies: Theoretical and Applied Problems, October 23- 25, 2024, Ternopil, Ukraine, Opole, Poland, 2024, pp. 486-493

УДК 681.518

А.М. Ковтко; В.Б. Савків, к.т.н., доц.; Г.В. Козбур, к.т.н., доц.; І.Р. Козбур
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АНАЛІЗ ЕФЕКТИВНОСТІ МУТАЦІЙНОГО ТЕСТУВАННЯ

A.M. Kovtko; V.B. Savkiv, Ph.D., Assoc. Prof.; H.V. Kozbur, Ph.D., Assoc. Prof.; I.R. Kozbur;

ANALYSIS OF THE EFFECTIVNES OF MUTATIONAL TESTING

Мутаційне тестування програмного забезпечення є перспективним і ефективним підходом до здійснення перевірки якості модульного тестування у сучасній розробці програмних інформаційних систем. Метою мутаційного тестування є оцінка здатності тестових наборів виявляти помилки шляхом створення штучних дефектів, тобто мутацій, у коді програми. Це дозволяє визначити повноту тестових сценаріїв та підвищити надійність, якість і стабільність програмного продукту. Важливість цієї теми зумовлена зростанням складності сучасних програмних продуктів, необхідністю підвищення їхньої якості та зменшення витрат на виправлення дефектів на пізніх етапах розробки. Ефективний аналіз результатів мутаційного тестування дозволяє оптимізувати процеси тестування, підвищити їх результативність та запроваджує більш високі стандарти якості програмного забезпечення.

Генерування модульних тестів є одним з ключових напрямків застосування великих мовних моделей (LLM) у сфері розробки програмного забезпечення [1]. Використання LLM у такому контексті дає змогу істотно скоротити тривалість написання тестів для програмного забезпечення та ефективніше розподілити ресурси команди розробників. Однак, попри ці переваги, згенеровані тести часто характеризуються недостатньою якістю, що зумовлено специфічними похибками («галюцинаціями») у самих мовних моделях [2]. Для подолання цієї проблеми доцільно проводити додатковий аналіз результатів, отриманих після генерації тестів. Авторами у [3, 4] запропоновано автоматизовану систему генерації модульних тестів. LLM генерує модульні тести, які підлягають контролю мутаційним тестуванням. Отримані результати вцілілих мутацій забезпечують уточнення та покращення початково згенерованих тестових випадків шляхом ітеративних звернень до LLM. Фактична реалізація зворотного зв'язку у системі тестування є очевидною ознакою її належності до автоматичних систем. Оскільки ефективність та якість згенерованих модульних тестів суттєво залежать від мутаційного тестування, актуальним завданням є дослідження рівня його впливу на кінцеву якість програмного продукту.

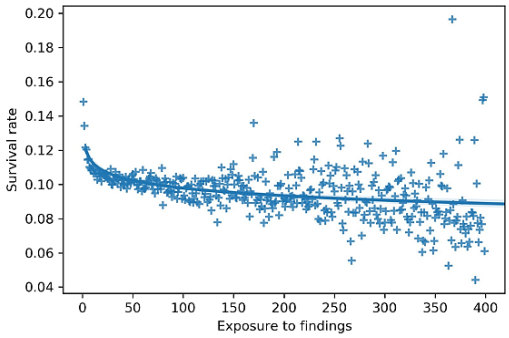

Рисунок 1. Кореляційна залежність рівня виживання мутантів від кількості введених мутацій

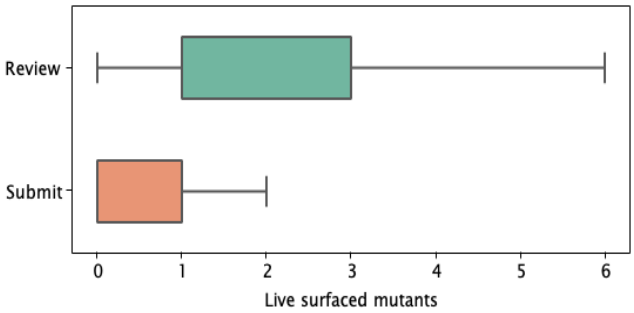

Рисунок 2. Кількість виявлених мутантів, які не виявлені на початку перевірки коду (Review) і наприкінці (Submit)

У роботі [5] авторами проведено аналіз взаємозв'язку між помилками, виявленими за допомогою мутаційного тестування, та реальними дефектами програмного забезпечення. Дослідження проведено на основі великого масиву емпіричних даних компанії Google, де за період у шість років було створено майже 15 мільйонів мутованих версій програмного коду в межах промислових проєктів. Результати показали, що у проєктах, які активно використовують мутаційне тестування, розробники створюють більшу кількість модульних тестів, що сприяє підвищенню рівня покриття коду. Крім того, тести, створені у відповідь на результати мутаційного аналізу, виявилися високоефективними, адже середня кількість невиявлених мутацій систематично зменшується з кожним наступним циклом тестування (Рис. 1). Показано також, що зміни, внесені в код після аналізу мутаційних звітів, суттєво знижують кількість невиявлених мутацій у подальших перевірках (Рис. 2). Важливим результатом стало й те, що тести, написані для виявлення конкретних мутацій, зазвичай дозволяють виявити також інші споріднені помилки. Встановлено прямий взаємозв'язок між мутаціями, які залишилися невиявленими після тестування, і реальними дефектами програмного забезпечення, що вказує на високу практичну ефективність мутаційного тестування як інструменту контролю якості програмного коду. Усі наведені результати підтверджують позитивний вплив мутаційного тестування на якість кінцевого програмного продукту незалежно від обраних технологій розробки. В розрізі системного підходу це підтверджує доцільність використання мутаційного тестування як джерела додаткового контексту для LLM [3, 4].

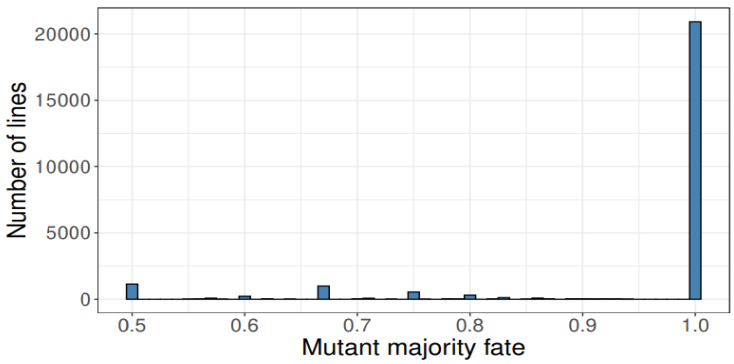

Рисунок 3. Результат виявлення мутантів для рядків із кількома мутантами

Слід зауважити, що система [3, 4] використовуватиме мутаційне тестування в ітераційному процесі, що посилює його головний недолік – час виконання мутаційного тестування. У [5] автори виявили, що генерація більше однієї мутації на рядок коду є надлишковою. Результати дослідження показали, що для більш ніж 90% рядків коду, для яких згенеровано дві та більше мутацій, помилки були виявлені за однією із них (Рис 3). Отже, генерація не більше однієї мутації на рядок коду є хорошою опцією оптимізації для пропонованої системи.

Література

1. Ozkaya, I. Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications / I. Ozkaya // IEEE Software. – 2023. – Vol. 40, No. 3. – P. 4–8.
2. Yan, L., Yan, C., Gao, S., Wang, W., Wang, B., Zhu, C., Chu, S., Zhou, J., Liang, G., Wang, Q., Chen, J. An Empirical Study on Unit Test Generation Using Large Language Models // arXiv. – 2024. – № 2406.18181.
3. Ковтко А. М. Автоматизована системи генерації модульних тестів на основі штучного інтелекту та мутаційного тестування / А. М. Ковтко, В. Б. Савків, І. Р. Козбур // Тези XIII МНПК „Актуальні задачі сучасних технологій“, 11-12 грудня 2024 року. – Т. : ФОП Паляниця В. А., 2024. – с. 417–418.
4. Ковтко, А., Савків, В. (2025). Розробка системи автоматизованого тестування на основі мутацій: переваги та виклики. Матеріали конференцій МЦНД, (31.01.2025; Дрогобич, Україна), 258–261. https://doi.org/10.62731/mcnd-31.01.2025.005.
5. Petrovic, G., Ivankovic, M., Fraser, G., Just, R. Does Mutation Testing Improve Testing Practices? / G. Petrovic, M. Ivankovic, G. Fraser, R. Just // Proceedings of the 43rd International Conference on Software Engineering (ICSE 2021). – 2021. – P. 910–921.

УДК 004.8:81'322

А. Конончук – ст. гр. СП-41, док. філософії. І. Мудрик
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

СТВОРЕННЯ УКРАЇНСЬКОЇ МОВНОЇ ГРИ НА ОСНОВІ EMBEDDING-МОДЕЛЕЙ

A. Kononchuk, Ph.D I. Mudryk

DEVELOPMENT OF A UKRAINIAN LANGUAGE GAME BASED ON EMBEDDING MODELS

У цій роботі описано процес створення української мовної гри «Словозв’яз», яка базується на визначенні семантичної близькості між словами. Ігровий процес побудований навколо введення користувачем слів, які порівнюються з цільовим словом за допомогою попередньо згенерованих embedding-представлень. Кожне слово в грі має контекстуально збагачене визначення, сформоване за допомогою GPT-моделі, що дозволяє точніше обчислити семантичну відстань між словами.

Семантичні представлення (embedding-и) для слів генеруються за допомогою Embedding API від OpenAI, що забезпечує високий рівень якості та точності при порівнянні значень. Технічна реалізація включає бекенд на Flask (Python), фронтенд на HTML/CSS/JS та зберігання попередньо обчислених даних у форматі JSON. Словникова база гри складається з 15 000 найуживаніших українських іменників, відібраних з частотного лексичного ресурсу UberText.

Генерація визначень для кожного слова за допомогою GPT API дозволила надати embedding-представленням необхідного контексту, що значно підвищило точність ранжування слів за змістовною подібністю. Згенеровані JSON-файли з ранжуванням зберігаються не лише у файловій системі, але також структуровано зберігаються в базі даних у вигляді архівних ігор. Це дозволяє забезпечити швидкий доступ до історичних даних та ефективно реалізувати щоденні та архівні ігри.

Гру розгорнуто на хостинговій платформі Render, де вона доступна для користувачів у відкритому доступі. Результати роботи демонструють можливість ефективного використання embedding-моделей для реалізації україномовних інтерактивних систем на основі смислового аналізу.

Література:

1. O. Bryk, I. Mudryk, M. Holubovskyi, Y Stoianov. Machine learning models and methods aspects of processing unstructured data. Proceedings of the 1st International Workshop on Bioinformatics and Applied Information Technologies (BAIT 2024) Zboriv, Ukraine, 2024. pp.