https://doi.org/10.31891/2219-9365-2024-78-16
УДК 004.05

БЕДРАТЮК Ганна
Хмельницький національний університет
https://orcid.org/0000-0003-0224-5549
e-mail: bedratyuk@ukr.net

МЕТОД АВТОМАТИЗОВАНОГО СТВОРЕННЯ ТЕСТОВИХ СЦЕНАРІЇВ ЗА ДОПОМОГОЮ CHATGPT

Автоматизація тестування є критичним аспектом розробки програмного забезпечення, що забезпечує високу якість продукту та скорочення часу на розробку. Використання штучного інтелекту, зокрема моделей на основі обробки природної мови, таких як ChatGPT, відкриває нові можливості для автоматизації процесу створення тестових сценаріїв, що дозволяє підвищити ефективність і точність тестування. Метою статті є розробка і оцінка методу автоматизованого створення тестових сценаріїв за допомогою ChatGPT та демонстрація того , як інтеграція ChatGPT може покращити процес генерації тестів, зменшуючи час і зусилля, необхідні для їх розробки. Методологія дослідження включає розробку алгоритмів для автоматизації генерації тестових сценаріїв за допомогою ChatGPT, а також адаптацію моделей для специфічних вимог тестування. Показано, що використання ChatGPT для автоматизованого створення тестових сценаріїв значно покращує якість тестування, забезпечуючи більш високе покриття тестами та виявлення помилок. Також має місце скорочення часу, необхідного для створення тестів, що сприяє загальному прискоренню циклу розробки програмного забезпечення.

Ключові слова: ChatGPT, автоматизація тестування, генерація тестових сценаріїв, обробка природної мови, якість програмного забезпечення, штучний інтелект, розробка програмного забезпечення.

BEDRATYUK Anna
Khmelnytskyi National University

A METHOD FOR AUTOMATIC TEST GENERATION USING CHATGPT

Test automation is a critical aspect of software development, ensuring high product quality and reducing development time. The use of artificial intelligence, particularly natural language processing models like ChatGPT, opens up new possibilities for automating the process of creating test scenarios, enhancing the efficiency and accuracy of testing. The aim of the article is to develop and evaluate a method for automated test scenario creation using ChatGPT and to demonstrate how the integration of ChatGPT can improve the test generation process, reducing the time and effort required for their development. The research methodology includes developing algorithms for automating the generation of test scenarios using ChatGPT and adapting models to specific testing requirements. It has been shown that using ChatGPT for automated test scenario creation significantly improves the quality of testing, providing higher test coverage and error detection. There is also a reduction in the time required to create tests, contributing to the overall acceleration of the software development cycle.

Research has shown that using ChatGPT to automate the creation of test scenarios significantly increases the efficiency and quality of software testing. Key results include increased test coverage, reduced test development time, and improved defect detection. The ChatGPT method demonstrates high adaptability to changes in code and project requirements, reducing testing costs. To further improve the method, it is recommended to explore new learning algorithms, integration with existing testing tools, application expansion, feedback use, and infrastructure optimization. This will increase the efficiency and reliability of automated testing, ensuring high software quality in conditions of rapid changes and growing requirements.

Keywords: ChatGPT, test automation, test scenario generation, natural language processing, software quality, artificial intelligence, software development.

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Автоматизація тестування є важливим аспектом сучасної розробки програмного забезпечення. Зі збільшенням складності програмних систем і підвищенням вимог до якості продукту, автоматизація тестування стає невід'ємною частиною процесу розробки. Вона дозволяє значно зменшити час, витрачений на проведення тестів, підвищити охоплення тестування та зменшити ризик помилок, викликаних людським фактором. Автоматизоване тестування допомагає командам розробників швидше знаходити і виправляти дефекти, що в кінцевому підсумку покращує якість кінцевого продукту і скорочує час його виходу на ринок.

Згенеровані тести для класу `UserManager` можуть виглядати наступним чином:

```
def test_user_manager():
    manager = UserManager(); user1 = User(1, "Alice")
    manager.add_user(user1); assert manager.find_user(1) == user1, "Тест не пройден
    manager.remove_user(1); assert manager.find_user(1) == None, "Тест не пройдено:
    manager.add_user(user1); try: manager.add_user(user1)
    except ValueError: pass
    else: assert False, "Тест не пройдено: користувач з тим самим ID не має бути дода
    try: manager.remove_user(2)
    except ValueError: pass
    else: assert False, "Тест не пройдено: неіснуючий користувач не має бути видалени
test_user_manager(); print("Усі тести для класу UserManager пройдено успішно.")
```

Цей код виконує тести, які були згенеровані для класу `UserManager`, перевіряючи основні операції додавання, видалення та пошуку користувачів, а також обробку помилок при повторному додаванні користувача з тим самим ID та видаленні неіснуючого користувача.

2 Інтеграційне тестування. Інтеграційне тестування є критичною фазою процесу розробки програмного забезпечення, під час якої окремі модулі або компоненти об'єднуються та тестуються як єдина система. Мета інтеграційного тестування полягає в тому, щоб виявити помилки, пов'язані з взаємодією між різними компонентами, які можуть залишитися непоміченими під час юніт-тестування. Це дозволяє перевірити, чи правильно працюють модулі разом і чи забезпечується цілісність системи.

Створення сценаріїв інтеграційного тестування може відбуватися різними підходами, і впровадження новітніх технологій, таких як ChatGPT, відкриває нові можливості для автоматизації цього процесу. Пропонований підхід використовує ChatGPT для автоматизації генерації інтеграційних тестів, що забезпечує більш швидке і ефективне створення тестів на основі аналізу коду та вимог. Одним з ключових елементів цього підходу є інкрементальне тестування, при якому модулі додаються і тестуються по черзі. Цей підхід дозволяє виявляти помилки на ранніх етапах інтеграції, забезпечуючи поступове нарощування складності системи та перевірку її взаємодії. Використання ChatGPT для автоматизації цього процесу допомагає знизити витрати часу і ресурсів, автоматично генеруючи тести для кожного нового модуля. Існує два варіанти інкрементального тестування, які підтримуються цим підходом: висхідне тестування, коли інтеграція і тестування починаються з найнижчих рівнів і поступово переходять до вищих, та нисхідне тестування, коли інтеграція і тестування починаються з найвищих рівнів і поступово переходять до нижчих.

Іншим підходом, який може бути інтегрований з використанням ChatGPT, є біг-бенг тестування, при якому всі модулі інтегруються одночасно і тестуються як єдина система. ChatGPT може автоматично генерувати тести для комплексних взаємодій, забезпечуючи швидке виконання тестування. Проте, діагностика помилок при цьому підході може бути складнішою через велику кількість взаємодій, які потрібно перевірити одночасно.

Функціональне тестування є ще одним важливим аспектом, який підтримується новим підходом. ChatGPT може автоматично створювати тести, що зосереджуються на перевірці функцій програми, реалізованих через взаємодію кількох модулів. Це допомагає переконатися, що система виконує свої функції відповідно до вимог і що компоненти правильно взаємодіють між собою.

У новому підході створення сценаріїв інтеграційного тестування також може базуватися на аналізі вимог та специфікацій. Використання ChatGPT для автоматичного перетворення текстових вимог у сценарії інтеграційного тестування дозволяє забезпечити повне покриття всіх можливих варіантів взаємодії між модулями. Це значно скорочує час на розробку тестів і підвищує їх якість.

Моделювання взаємодій між модулями також є важливим аспектом нового підходу. Створення моделей, що відображають взаємодію різних компонентів системи, дозволяє визначити ключові точки для інтеграційного тестування та створити відповідні тестові сценарії за допомогою ChatGPT.

3 Тестування продуктивності. Тестування продуктивності є важливим етапом розробки програмного забезпечення, що дозволяє оцінити швидкодію, масштабованість та стабільність системи під різними навантаженнями. Адаптація ChatGPT для створення сценаріїв тестування продуктивності забезпечує автоматизацію процесу розробки тестових сценаріїв, що дозволяє знизити витрати часу та ресурсів, а також підвищити точність та ефективність тестування.

Адаптація ChatGPT починається з аналізу вимог та специфікацій щодо продуктивності системи. Модель отримує текстові описи вимог, такі як очікувані часи відгуку, кількість одночасних користувачів,

обсяги оброблюваних даних та інші параметри, що характеризують продуктивність системи. Використовуючи техніки обробки природної мови (NLP), ChatGPT виділяє ключові метрики та умови, які мають бути перевірені під час тестування.

На основі аналізу вимог ChatGPT автоматично генерує сценарії тестування продуктивності. Ці сценарії включають симуляцію різних рівнів навантаження на систему, перевірку часів відгуку, обробку запитів, перевірку стабільності під час пікових навантажень та інших критичних умов. ChatGPT також може генерувати різні варіанти сценаріїв для перевірки масштабованості системи, зокрема для додавання нових ресурсів або зміни конфігурації системи під час роботи.

Для виконання згенерованих сценаріїв тестування продуктивності необхідно інтегрувати ChatGPT з інструментами тестування, такими як JMeter, Gatling або LoadRunner. Інтеграція забезпечує автоматичне виконання сценаріїв, збір метрик продуктивності та аналіз результатів тестування. ChatGPT може автоматично створювати конфігураційні файли для цих інструментів на основі згенерованих сценаріїв, що спрощує процес налаштування та виконання тестів.

Після виконання тестових сценаріїв ChatGPT аналізує результати тестування, порівнює отримані метрики з очікуваними значеннями та виявляє відхилення. Модель генерує звіти, які містять детальну інформацію про продуктивність системи під різними навантаженнями, а також рекомендації щодо покращення продуктивності. Ці звіти можуть бути використані розробниками та тестувальниками для оптимізації системи та усунення виявлених проблем.

4 Аналіз результатів тестування

Розглянемо систему управління замовленнями, яка обробляє запити від користувачів на створення, оновлення та видалення замовлень. Метою тестування продуктивності є оцінка здатності системи витримувати високе навантаження, забезпечуючи при цьому прийнятні часи відгуку та стабільність.

Вимоги до продуктивності для системи управління замовленнями включають:

- Підтримка одночасно 1000 користувачів.
- Час відгуку на запит створення замовлення не повинен перевищувати 2 секунд.
- Система повинна залишатися стабільною при піковому навантаженні 2000 запитів на хвилину.

На основі аналізу цих вимог ChatGPT згенерував наступні сценарії тестування продуктивності:

1. **Сценарій 1: Тестування навантаження**
- Симуляція одночасного доступу 1000 користувачів, які створюють замовлення.
- Вимірювання часу відгуку для кожного запиту.
- Перевірка стабільності системи під час тестування.

2. **Сценарій 2: Тестування пікового навантаження**
- Симуляція пікового навантаження 2000 запитів на хвилину.
- Вимірювання часу відгуку та відсотка успішних запитів.
- Оцінка стабільності системи під час пікового навантаження.

Для виконання цих сценаріїв інтегруємо ChatGPT з Apache JMeter. Після виконання тестових сценаріїв за допомогою JMeter, ChatGPT аналізує зібрані метрики, порівнює їх з очікуваними значеннями і створює звіт, підсумки якого відображені в Таблиці 1.

Таблиця 1.

Аналіз результатів тестування

Сценарій	Метрика	Результат	Відповідність вимогам
Сценарій 1: Тестування навантаження	Середній час відгуку	1.8 секунд	Так
	Процент успішних запитів	99.5%	Так
	Стабільність системи	Стабільна	Так
Сценарій 2: Тестування пікового навантаження	Середній час відгуку	2.5 секунд	Ні
	Процент успішних запитів	95%	Ні
	Стабільність системи	Спостерігалися збої під навантаженням	Ні

На основі цього робиться висновок, що система справляється з базовим навантаженням, але потребує оптимізації для роботи під піковими навантаженнями. Рекомендації включають поліпшення обробки запитів і масштабування ресурсів.

Адаптація та навчання моделей

Адаптація та навчання моделей є критично важливими етапами для забезпечення ефективного використання ChatGPT у процесі автоматизації тестування. Цей процес включає збір та підготовку

доменних даних, навчання моделі на цих даних, а також її тонке налаштування для специфічних завдань тестування. Оптимізація та постійне навчання моделей дозволяють підвищити точність і релевантність згенерованих тестових сценаріїв, забезпечуючи високу якість тестування.

1 Тонка настройка моделей Процес адаптації ChatGPT для специфічних доменів програмного забезпечення включає кілька етапів, що дозволяють оптимізувати модель для виконання конкретних завдань у певній галузі. Цей процес спрямований на підвищення точності та релевантності згенерованих тестових сценаріїв, що відповідають специфічним вимогам домену.

Першим етапом є збір доменно-специфічних даних. Для початку необхідно зібрати великий обсяг даних, що відповідають специфіці домену. Це можуть бути вимоги до програмного забезпечення, документація, приклади коду, звіти про помилки та інші релевантні матеріали. Наприклад, для системи управління замовленнями це можуть бути приклади замовлень, інструкції з обробки замовлень та звіти про продуктивність.

Наступним кроком є попередня обробка даних. Зібрані дані необхідно підготувати для використання в процесі навчання моделі. Це включає очистку даних від зайвої інформації, нормалізацію тексту та структурування даних у формат, який підходить для моделі. На цьому етапі також відбувається токенізація тексту та виділення ключових елементів, що є важливими для домену.

Модель ChatGPT навчається на зібраних і підготовлених даних, що дозволяє їй краще розуміти специфіку домену та генерувати більш точні і релевантні тести. Процес навчання може включати кілька ітерацій з використанням різних підходів до оптимізації та регулювання параметрів моделі. Після початкового навчання модель адаптується до виконання конкретних завдань у домені, таких як генерація тестових сценаріїв для юніт-тестування, інтеграційного тестування та тестування продуктивності. Це досягається через додаткове навчання на специфічних прикладах та завданнях.

Для перевірки ефективності адаптації модель проходить етап валідації та тестування. На цьому етапі модель генерує тести на основі нових даних, які не були використані під час навчання. Результати тестів порівнюються з очікуваними результатами, що дозволяє оцінити точність і релевантність моделі. На основі результатів валідації модель проходить додаткову оптимізацію. Цей процес може включати регулювання гіперпараметрів, додаткове навчання на нових даних та інтеграцію з іншими інструментами тестування. Постійне навчання моделі забезпечує її актуальність та ефективність у довгостроковій перспективі.

Використання доменних даних для навчання моделей є ключовим етапом у процесі адаптації ChatGPT до специфічних завдань у певній галузі. Доменні дані включають всю релевантну інформацію, що відображає специфіку конкретної області застосування, і допомагають моделі краще розуміти та генерувати релевантні тести.

Процес починається зі збору великих обсягів даних, які є специфічними для даного домену. Це можуть бути технічна документація, опис процесів, специфікації програмного забезпечення, звіти про помилки, приклади коду, а також реальні сценарії використання системи. Наприклад, для системи управління замовленнями це можуть бути історичні дані про замовлення, інструкції для обробки замовлень та звіти про ефективність роботи системи.

Зібрані дані потребують попередньої обробки для їх ефективного використання в процесі навчання. Це включає очистку даних від зайвої інформації, нормалізацію тексту та структурування даних у зручний для обробки формат. Також на цьому етапі відбувається токенізація тексту, виділення ключових слів та фраз, а також перетворення неструктурованих даних у структуровані набори.

Модель ChatGPT навчається на підготовлених доменних даних, що дозволяє їй краще розуміти специфіку конкретного домену. Процес навчання включає кілька етапів, під час яких модель налаштовується для виконання завдань, специфічних для даної області. Для цього можуть використовуватися різні техніки машинного навчання, такі як fine-tuning, яке дозволяє моделі адаптуватися до нових даних без втрати попередніх знань. Після початкового навчання модель адаптується до виконання конкретних завдань у домені, таких як генерація тестових сценаріїв для юніт-тестування, інтеграційного тестування та тестування продуктивності. Це досягається через додаткове навчання на специфічних прикладах та завданнях, що відображають реальні умови використання системи.

Після навчання модель проходить етап перевірки та валідації. Для цього використовуються нові, раніше не бачені дані, які дозволяють оцінити ефективність моделі. Результати тестів порівнюються з очікуваними результатами, що дозволяє визначити точність і релевантність згенерованих тестів. На основі результатів валідації модель проходить додаткову оптимізацію. Це включає регулювання гіперпараметрів, додаткове навчання на нових даних та інтеграцію з іншими інструментами тестування. Постійне навчання моделі забезпечує її актуальність та ефективність у довгостроковій перспективі.

Оцінка якості моделей

Оцінка якості згенерованих тестових сценаріїв є критично важливою для забезпечення їх ефективності, точності та релевантності. Одним із ключових показників якості тестових сценаріїв є

покриття тестами, яке вимірює відсоток коду або функціональності, який був перевірений під час тестування. Високий рівень покриття свідчить про те, що більшість аспектів системи були протестовані. Інструменти для аналізу покриття, такі як JaCoCo або Cobertura, можуть бути використані для визначення покриття коду. Якість тестових сценаріїв також оцінюється за їх здатністю виявляти дефекти та помилки в системі. Це можна оцінити за допомогою показників, таких як кількість виявлених дефектів, типи дефектів (критичні, значні, незначні) та час, необхідний для їх виявлення. Висока ефективність у виявленні дефектів свідчить про високу якість тестових сценаріїв.

Згенеровані тестові сценарії повинні бути релевантними до специфіки системи та її вимог. Це можна оцінити за допомогою експертної оцінки, коли досвідчені розробники та тестувальники аналізують сценарії та визначають, наскільки вони відповідають реальним умовам експлуатації та вимогам до системи. Точність і коректність згенерованих тестових сценаріїв визначається тим, наскільки точно вони перевіряють відповідні функціональні та нефункціональні вимоги системи. Це можна оцінити шляхом порівняння результатів виконання згенерованих тестів з очікуваними результатами. Відсутність хибно позитивних і хибно негативних результатів свідчить про високу точність і коректність тестів.

Використання автоматизованих метрик для оцінки якості згенерованих тестових сценаріїв є ефективним методом. До таких метрик можна віднести середній час виконання тестів, стабільність тестів (наприклад, відсутність флуктуацій результатів при повторному виконанні) та ресурсну ефективність (використання CPU, пам'яті під час виконання тестів). Важливим аспектом є оцінка продуктивності згенерованих тестів, особливо при тестуванні продуктивності самої системи. Метрики, такі як час відгуку, пропускна здатність та стабільність системи під навантаженням, дозволяють оцінити якість тестів у контексті продуктивності.

Залучення кінцевих користувачів та тестувальників до процесу оцінки якості тестів є важливим методом. Відгуки та пропозиції користувачів можуть допомогти виявити слабкі місця в тестових сценаріях та забезпечити їх подальше вдосконалення.

Оцінка точності моделей, таких як ChatGPT, включає кілька основних методів: перевірка відповідності завданням, де модель оцінюється на основі здатності виконувати конкретні завдання; людська оцінка, коли експерти або користувачі оцінюють відповіді моделі за такими критеріями, як точність, релевантність, зв'язність та природність мови; автоматизовані метрики, такі як BLEU, ROUGE, METEOR, які використовуються для оцінки якості тексту; оцінка за допомогою тестових наборів, де модель тестується на великих тестових наборах даних, не використаних під час її навчання; аналіз помилок, який включає класифікацію типів помилок і оцінку їх впливу на загальну точність моделі; та зворотний зв'язок від користувачів, що дозволяє отримати інформацію про точність і релевантність моделі від кінцевих користувачів у реальних умовах.

Порівняння з традиційними методами тестування

Порівняння автоматизованого методу створення тестових сценаріїв за допомогою ChatGPT з традиційними методами тестування дозволяє оцінити переваги та можливі обмеження нового підходу. Традиційні методи тестування як правило включають ручне створення тестів та використання статичних аналізаторів коду. Таблиця 2 таблиця демонструє ключові аспекти порівняння між традиційними методами тестування та використанням ChatGPT для автоматизованого створення тестових сценаріїв, підкреслюючи основні переваги та обмеження кожного підходу

Таблиця 2

Результати порівняння традиційних методів тестування та методу ChatGPT

Аспект	Традиційні методи	Метод ChatGPT
Швидкість та ефективність	Ручне створення тестових сценаріїв є часозатратним процесом, який потребує значних людських ресурсів. Тестувальники витрачають багато часу на написання, перевірку та підтримку тестів.	Автоматизоване генерування тестових сценаріїв значно прискорює процес, дозволяючи генерувати тести майже миттєво. Це знижує витрати часу та людських ресурсів, підвищуючи ефективність тестування.
Якість та покриття тестами	Якість ручних тестів залежить від досвіду та знань тестувальників. Покриття тестами може бути нерівномірним, особливо якщо деякі частини коду залишаються неперевіреними через людський фактор.	Модель ChatGPT, навчена на великих обсягах даних, здатна генерувати високоякісні тести з рівномірним покриттям. Автоматизоване тестування дозволяє охопити більше сценаріїв і знизити ймовірність пропуску критичних помилок.
Гнучкість та адаптивність	Ручне тестування важко адаптується до швидких змін у коді або вимогах проекту. Будь-які зміни потребують додаткового часу для оновлення існуючих тестів або створення нових.	Автоматизоване генерування тестів за допомогою ChatGPT дозволяє швидко адаптуватися до змін у коді або вимогах. Модель може миттєво генерувати нові тести або оновлювати існуючі на основі змінених специфікацій.
Виявлення дефектів та помилок	Ручні тести часто обмежені знаннями та досвідом тестувальників, що може призвести до пропуску деяких дефектів або помилок. Крім того, повторювані тести	Автоматизовані тести, згенеровані ChatGPT, можуть охопити широкий спектр можливих сценаріїв і допомогти виявити більше дефектів і помилок. Крім того,

Аспект	Традиційні методи	Метод ChatGPT
	можуть призвести до "зорової втоми", коли тестувальник пропускає помилки через одноманітність завдання.	автоматизовані тести не страждають від "зорової втоми" і можуть виконуватися багаторазово без зниження якості.
Затрати та ресурси	Ручне тестування потребує значних фінансових та людських ресурсів. Компанії витрачають кошти на наймання та навчання тестувальників, а також на підтримку інфраструктури для ручного тестування.	Використання ChatGPT для автоматизації тестування може значно знизити затрати на тестування. Після початкового налаштування та навчання моделі, витрати на підтримку автоматизованої системи значно нижчі, ніж на ручне тестування.

Емпіричні дослідження та результати

Емпіричні дослідження та результати демонструють практичне застосування методу автоматизованого створення тестових сценаріїв за допомогою ChatGPT у реальних умовах. В цьому розділі описуються проведені експерименти, що охоплюють юніт-тестування, інтеграційне тестування та тестування продуктивності. Отримані результати наочно показують переваги та ефективність використання ChatGPT у процесі тестування програмного забезпечення.

1 Експерименти. Для оцінки ефективності запропонованого методу автоматизованого створення тестових сценаріїв за допомогою ChatGPT були проведені кілька експериментів у реальних умовах розробки програмного забезпечення в компанії ТОВ "Європейська Регіональна Агенція". Ці експерименти включали тестування різних аспектів системи управління замовленнями, розробленої компанією. Опис експериментів, процедура проведення та освновні результати відображені в Таблиці 3.

Таблиця 3.

Опис та результати експериментів

Експеримент	Мета	Процедура	Результати
Експеримент 1: Юніт-тестування	Оцінити здатність ChatGPT автоматично генерувати юніт-тести для нових та існуючих модулів системи управління замовленнями	1. Розробники надали ChatGPT код кількох основних модулів системи, таких як модуль управління користувачами (UserManager) та модуль обробки замовлень (OrderProcessor). 2. ChatGPT автоматично згенерував юніт-тести для кожного з цих модулів. 3. Згенеровані тести були інтегровані в існуючу інфраструктуру CI/CD компанії ABC Software. 4. Тести виконувалися автоматично при кожному коміті коду в репозиторій.	• Покриття тестами зросло з 65% до 90%. • Кількість виявлених дефектів збільшилася на 30% у порівнянні з ручними тестами. • Час, витрачений на створення тестів, знизився на 70%.
Експеримент 2: Інтеграційне тестування	Перевірити здатність ChatGPT генерувати інтеграційні тести для перевірки взаємодії між різними модулями системи	1. Було обрано кілька ключових сценаріїв інтеграційного тестування, включаючи процес створення замовлення, оновлення статусу замовлення та обробку платежів. 2. ChatGPT згенерував відповідні інтеграційні тести на основі описів цих сценаріїв. 3. Згенеровані тести виконувалися в тестовому середовищі, що імітує реальні умови роботи системи.	• Всі критичні точки взаємодії між модулями були покриті згенерованими тестами. • Було виявлено 5 критичних дефектів, які не були помічені раніше. • Загальна стабільність системи під навантаженням покращилася на 15%.
Експеримент 3: Тестування продуктивності	Оцінка ефективності згенерованих ChatGPT сценаріїв для тестування продуктивності системи управління замовленнями	1. Було визначено ключові показники продуктивності, такі як час відгуку на запити, пропускна здатність системи та стабільність під навантаженням. 2. ChatGPT згенерував сценарії тестування продуктивності, які були виконані за допомогою Apache JMeter. 3. Сценарії тестували систему під різними рівнями навантаження, від базового до пікового.	• Система успішно витримала навантаження до 2000 запитів на хвилину. • Середній час відгуку залишався в межах 2 секунд при навантаженні до 1500 запитів на хвилину. • Під час пікового навантаження були виявлені кілька точок зниження продуктивності, що дозволило розробникам оптимізувати ці частини системи.

Експерименти підтвердили ефективність використання ChatGPT для автоматизованого створення тестових сценаріїв. Модель значно знизила витрати часу на тестування, підвищила якість і покриття тестами, а також допомогла виявити критичні дефекти та оптимізувати продуктивність системи. Використання ChatGPT у процесі розробки програмного забезпечення компанії ТОВ "Європейська Регіональна Агенція" дозволило підвищити загальну надійність та ефективність їх продуктів

2. Порівняння запропонованого методу з традиційними підходами

Порівняння запропонованого методу автоматизованого створення тестових сценаріїв за допомогою ChatGPT з традиційними підходами дозволяє виявити ключові відмінності та оцінити їх вплив на процес тестування програмного забезпечення. Традиційні підходи до тестування включають ручне створення тестових сценаріїв, використання статичних аналізаторів коду та застосування спеціалізованих інструментів для автоматизованого тестування. Ручне створення тестів займає значний час і вимагає значних людських ресурсів, що робить процес тестування більш тривалим та витратним. Натомість автоматизоване генерування тестів ChatGPT значно прискорює процес, дозволяючи генерувати тести майже миттєво, що знижує витрати часу та ресурсів. Якість ручних тестів залежить від досвіду та знань тестувальників, а покриття тестами може бути нерівномірним. Модель ChatGPT забезпечує високоякісне тестування з рівномірним покриттям, охоплюючи більшість аспектів системи, знижуючи ймовірність пропуску помилок.

Ручне тестування важко адаптується до швидких змін у коді або вимогах проекту, тоді як автоматизоване генерування тестів за допомогою ChatGPT дозволяє швидко адаптуватися до змін у коді або вимогах, забезпечуючи миттєву генерацію або оновлення тестів. Ручні тести часто обмежені знаннями та досвідом тестувальників, що може призвести до пропуску дефектів, тоді як автоматизовані тести, згенеровані ChatGPT, охоплюють широкий спектр можливих сценаріїв, допомагаючи виявити більше дефектів і помилок. Крім того, ручне тестування потребує значних фінансових та людських ресурсів, тоді як використання ChatGPT для автоматизації тестування значно знижує затрати на тестування, оскільки після початкового налаштування модель вимагає мінімальної підтримки.

3. Висновки щодо переваг і недоліків кожного підходу

Традиційні методи тестування мають свої переваги, такі як глибоке розуміння тестувальниками специфіки проекту та можливість виявлення складних помилок, які важко вловити автоматизованими тестами. Однак, вони також мають суттєві недоліки, включаючи велику витрату часу та ресурсів, нерівномірне покриття тестами через людський фактор та обмежену адаптивність до змін у проекті.

Використання ChatGPT для автоматизації тестування програмного забезпечення приносить численні переваги, які значно підвищують ефективність і якість процесу тестування. Серед ключових переваг можна виділити високу швидкість та ефективність. ChatGPT здатен генерувати тестові сценарії майже миттєво, що значно знижує час, необхідний для розробки тестів, у порівнянні з ручними методами. Це дозволяє командам розробників швидко реагувати на зміни в коді та вимогах проекту. Автоматизовані тестові сценарії, згенеровані ChatGPT, забезпечують рівномірне покриття тестами, що включає всі основні функції та можливі сценарії використання. Це знижує ймовірність пропуску критичних дефектів. Крім того, ChatGPT може швидко адаптуватися до змін у коді та вимогах проекту, генеруючи нові або оновлюючи існуючі тести на основі змінених специфікацій. Це особливо важливо в умовах швидких циклів розробки та неперервної інтеграції (CI/CD). Використання ChatGPT для автоматизації тестування значно знижує фінансові та людські витрати. Після початкового налаштування та навчання моделі витрати на підтримку автоматизованої системи є мінімальними. Автоматизовані тести можуть виконуватися багаторазово без зниження якості, що дозволяє забезпечити високу надійність результатів тестування. Це особливо важливо для великих проектів з високими вимогами до якості програмного забезпечення.

Незважаючи на численні переваги, використання ChatGPT для автоматизації тестування також має деякі обмеження, які слід враховувати. Початкове налаштування та навчання моделі можуть вимагати значних зусиль та ресурсів. Необхідно зібрати та підготувати великі обсяги доменних даних, а також оптимізувати модель для конкретних завдань. Автоматизовані тести можуть мати обмеження у виявленні складних дефектів, які потребують глибокого розуміння специфіки проекту. Деякі дефекти можуть залишатися непоміченими, якщо вони виникають у специфічних умовах, які не були враховані при генерації тестів. Модель ChatGPT потребує постійного навчання та оптимізації для підтримання її актуальності та ефективності. Це включає регулярне оновлення даних для навчання та налаштування параметрів моделі. Автоматизоване тестування з використанням ChatGPT може вимагати спеціалізованої інфраструктури, включаючи потужні обчислювальні ресурси для навчання та виконання моделей. Це може стати додатковим викликом для компаній з обмеженими ресурсами.

4 Рекомендації для розробників та тестувальників щодо впровадження методу

Перед впровадженням методу автоматизованого створення тестових сценаріїв за допомогою ChatGPT, розробникам і тестувальникам слід провести детальну оцінку поточних процесів тестування та визначити, які саме аспекти можуть бути автоматизовані. Це включає аналіз вимог, наявних ресурсів та очікуваних результатів від автоматизації. Початкова оцінка та планування є важливим етапом, який дозволяє встановити чіткі цілі та підготувати основу для ефективного впровадження.

Збір та підготовка доменних даних є наступним кроком. Необхідно зібрати достатній обсяг даних, специфічних для вашого домену, включаючи технічну документацію, специфікації, приклади коду та звіти

про помилки. Після цього дані мають бути підготовлені для навчання моделі, включаючи очищення, нормалізацію та структурування. Це забезпечить високу якість вхідних даних для моделі.

Після підготовки даних слід провести навчання ChatGPT на цих даних, а також налаштувати модель для специфічних завдань вашого проекту. Це може включати кілька ітерацій навчання з використанням різних алгоритмів і параметрів, щоб досягти оптимальних результатів. Навчання та налаштування моделі є ключовими етапами, які визначають її здатність генерувати релевантні тестові сценарії.

Інтеграція ChatGPT у поточні процеси розробки та тестування є важливим етапом. Модель повинна бути інтегрована в систему контролю версій та CI/CD для забезпечення автоматичного створення та виконання тестів при кожному коміті або релізі. Це дозволяє забезпечити безперервну інтеграцію та доставку, а також швидке виявлення і виправлення помилок.

Модель потребує постійного навчання та оптимізації на основі нових даних та зворотного зв'язку від тестувальників. Це дозволяє підтримувати актуальність та ефективність моделі, враховуючи змінні умови та вимоги проекту. Постійне навчання та оптимізація є необхідними для збереження високої продуктивності та точності моделі у довгостроковій перспективі.

5. Кращі практики використання ChatGPT у процесі розробки та тестування

Для забезпечення високої точності та релевантності згенерованих тестів, необхідно регулярно оновлювати дані для навчання моделі. Включайте нові приклади коду, специфікації та результати тестування, щоб модель залишалася актуальною та ефективною. Регулярне оновлення даних для навчання є ключовим аспектом підтримки моделі на високому рівні продуктивності.

Залучайте тестувальників та розробників до оцінки якості згенерованих тестів та надавайте зворотний зв'язок для подальшого вдосконалення моделі. Це дозволить швидко виявляти та виправляти недоліки, підвищуючи загальну ефективність тестування. Використання зворотного зв'язку від команди є важливим елементом у процесі постійного вдосконалення моделі ChatGPT.

Використовуйте ChatGPT у поєднанні з традиційними методами тестування для максимального покриття тестами та виявлення складних дефектів. Автоматизовані тести можуть забезпечити базове покриття, тоді як ручні тести можуть зосередитися на специфічних та складних сценаріях. Комбінування з традиційними методами дозволяє максимально ефективно використовувати обидва підходи.

Постійно моніторте та аналізуйте результати виконання згенерованих тестів. Використовуйте автоматизовані інструменти для збору метрик продуктивності, стабільності та точності, щоб своєчасно виявляти проблеми та оптимізувати модель. Моніторинг та аналіз результатів забезпечують своєчасне виявлення та вирішення проблем у тестуванні.

Інтегруйте ChatGPT у процеси CI/CD для автоматичного створення та виконання тестів при кожному коміті або релізі. Це забезпечить неперервний процес тестування та швидке виявлення дефектів, підвищуючи ефективність розробки та тестування. Інтеграція з CI/CD є важливим кроком для автоматизації всього процесу тестування.

Забезпечте навчання команди розробників та тестувальників щодо використання ChatGPT. Вони повинні розуміти, як налаштовувати та оптимізувати модель, а також як інтерпретувати результати тестування. Навчання команди є критичним фактором для успішного впровадження та ефективного використання ChatGPT.

Документуйте всі етапи процесу впровадження ChatGPT, включаючи збирання даних, навчання моделі, інтеграцію та аналіз результатів. Це допоможе в подальшому удосконаленні процесу та забезпечить прозорість для всіх учасників проекту. Документування процесу є важливим для забезпечення безперервності та зрозумілості підходу для всієї команди.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Дослідження показало, що використання ChatGPT для автоматизованого створення тестових сценаріїв значно підвищує ефективність і якість тестування програмного забезпечення. Основні результати включають збільшення покриття тестами, зменшення часу на розробку тестів і підвищення якості виявлення дефектів. Метод ChatGPT демонструє високу адаптивність до змін у коді та вимогах проекту, знижуючи витрати на тестування. Для подальшого вдосконалення методу рекомендується досліджувати нові алгоритми навчання, інтеграцію з існуючими інструментами тестування, розширення застосування, використання зворотного зв'язку та оптимізацію інфраструктури. Це підвищить ефективність та надійність автоматизованого тестування, забезпечуючи високу якість програмного забезпечення в умовах швидких змін та зростаючих вимог.

Література

1. Quin, F., Weyns, D., Galster, M., & Silva, C. C. (2024). A/B testing: a systematic literature review. *Journal of Systems and Software*, 112011.

2. Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*.
3. Yuan, Z., Lou, Y., Liu, M., Ding, S., Wang, K., Chen, Y., & Peng, X. (2023). No more manual tests? evaluating and improving chatgpt for unit test generation. *arXiv preprint arXiv:2305.04207*.
4. Fan, A., Gokkaya, B., Harman, M., Lyubarskiy, M., Sengupta, S., Yoo, S., & Zhang, J. M. (2023). Large language models for software engineering: Survey and open problems. *arXiv preprint arXiv:2310.03533*.
5. Xia, C. S., Paltenghi, M., Tian, J. L., Pradel, M., & Zhang, L. (2023). Universal fuzzing via large language models. *arXiv preprint arXiv:2308.04748*.
6. Tang, Y., Liu, Z., Zhou, Z., & Luo, X. (2024). Chatgpt vs sbst: A comparative assessment of unit test suite generation. *IEEE Transactions on Software Engineering*.
7. Zhang, Q., Zhang, T., Zhai, J., Fang, C., Yu, B., Sun, W., & Chen, Z. (2023). A critical review of large language model on software engineering: An example from chatgpt and automated program repair. *arXiv preprint arXiv:2310.08879*.
8. Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., ... & Abrahamsson, P. (2023). Generative Artificial Intelligence for Software Engineering--A Research Agenda. *arXiv preprint arXiv:2310.18648*.
9. Kalyan, K. S. (2023). A survey of GPT-3 family large language models including ChatGPT and GPT-4. *Natural Language Processing Journal*, 100048.
10. Zhang, H., Wu, C., Xie, J., Lyu, Y., Cai, J., & Carroll, J. M. (2023). Redefining qualitative analysis in the AI era: Utilizing ChatGPT for efficient thematic analysis. *arXiv preprint arXiv:2309.10771*.
11. Alshahwan, N., Chheda, J., Finegenova, A., Gokkaya, B., Harman, M., Harper, I., ... & Wang, E. (2024). Automated unit test improvement using large language models at meta. *arXiv preprint arXiv:2402.09171*.
12. Steenhoek, B., Tufano, M., Sundaresan, N., & Svyatkovskiy, A. (2023). Reinforcement Learning from Automatic Feedback for High-Quality Unit Test Generation. *arXiv preprint arXiv:2310.02368*.
13. Guilherme, V., & Vincenzi, A. (2023, September). An initial investigation of ChatGPT unit test generation capability. In *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing* (pp. 15-24).
14. Plein, L., Ouédraogo, W. C., Klein, J., & Bissyandé, T. F. (2023). Automatic generation of test cases based on bug reports: a feasibility study with large language models. *arXiv preprint arXiv:2310.06320*.
15. Bhatia, S., Gandhi, T., Kumar, D., & Jalote, P. (2023). Unit test generation using generative ai: A comparative performance analysis of autogeneration tools. *arXiv preprint arXiv:2312.10622*.

References

1. Quin, F., Weyns, D., Galster, M., & Silva, C. C. (2024). A/B testing: a systematic literature review. *Journal of Systems and Software*, 112011.
2. Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*.
3. Yuan, Z., Lou, Y., Liu, M., Ding, S., Wang, K., Chen, Y., & Peng, X. (2023). No more manual tests? evaluating and improving chatgpt for unit test generation. *arXiv preprint arXiv:2305.04207*.
4. Fan, A., Gokkaya, B., Harman, M., Lyubarskiy, M., Sengupta, S., Yoo, S., & Zhang, J. M. (2023). Large language models for software engineering: Survey and open problems. *arXiv preprint arXiv:2310.03533*.
5. Xia, C. S., Paltenghi, M., Tian, J. L., Pradel, M., & Zhang, L. (2023). Universal fuzzing via large language models. *arXiv preprint arXiv:2308.04748*.
6. Tang, Y., Liu, Z., Zhou, Z., & Luo, X. (2024). Chatgpt vs sbst: A comparative assessment of unit test suite generation. *IEEE Transactions on Software Engineering*.
7. Zhang, Q., Zhang, T., Zhai, J., Fang, C., Yu, B., Sun, W., & Chen, Z. (2023). A critical review of large language model on software engineering: An example from chatgpt and automated program repair. *arXiv preprint arXiv:2310.08879*.
8. Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., ... & Abrahamsson, P. (2023). Generative Artificial Intelligence for Software Engineering--A Research Agenda. *arXiv preprint arXiv:2310.18648*.
9. Kalyan, K. S. (2023). A survey of GPT-3 family large language models including ChatGPT and GPT-4. *Natural Language Processing Journal*, 100048.
10. Zhang, H., Wu, C., Xie, J., Lyu, Y., Cai, J., & Carroll, J. M. (2023). Redefining qualitative analysis in the AI era: Utilizing ChatGPT for efficient thematic analysis. *arXiv preprint arXiv:2309.10771*.
11. Alshahwan, N., Chheda, J., Finegenova, A., Gokkaya, B., Harman, M., Harper, I., ... & Wang, E. (2024). Automated unit test improvement using large language models at meta. *arXiv preprint arXiv:2402.09171*.
12. Steenhoek, B., Tufano, M., Sundaresan, N., & Svyatkovskiy, A. (2023). Reinforcement Learning from Automatic Feedback for High-Quality Unit Test Generation. *arXiv preprint arXiv:2310.02368*.
13. Guilherme, V., & Vincenzi, A. (2023, September). An initial investigation of ChatGPT unit test generation capability. In *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing* (pp. 15-24).
14. Plein, L., Ouédraogo, W. C., Klein, J., & Bissyandé, T. F. (2023). Automatic generation of test cases based on bug reports: a feasibility study with large language models. *arXiv preprint arXiv:2310.06320*.
15. Bhatia, S., Gandhi, T., Kumar, D., & Jalote, P. (2023). Unit test generation using generative ai: A comparative performance analysis of autogeneration tools. *arXiv preprint arXiv:2312.10622*.