УДК 004.8

Ворона О.О.
Васильєв К.О., к.т.н.
Полтавський національний технічний
університет імені Юрія Кондратюка

АНАЛІЗ АВТОМАТИЗАЦІЇ ОКРЕМИХ ЕТАПІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Анотація. *У роботі проведено аналіз автоматизації етапів розробки програмного забезпечення з використанням інтелектуальних систем. Було проаналізовано використання інтелектуальних систем в етапах тестування та реалізації додатків.*

Ключові слова*: тестування, програмне забезпечення, предиктивна аналітика, машинне навчання.*

Вступ

У вік інформаційних технологій люди прагнуть автоматизувати якомога більшу частину свого життя, і галузь розробки програмного забезпечення (ПЗ) не є виключенням. Зрозуміло, що певні етапи розробки ПЗ не можна представити без втручання людини через наявність творчої або організаційної частин. Але деякі з них все ж таки слугують гарним плацдармом для застосування інтелектуальних систем з метою отримання вигоди від автоматизації.

Програми для автоматичного тестування готового продукту мають досить високу цінність. Даний спосіб тестування набув значного розповсюдження останнім часом і є досить ефективним через свою предикативну природу – программа у ході тестування навчається та дозволяє перевірити різноманітні варіанти поведінки гри. А найважливішим є те, що під час тестування

найбільшою мірою охоплюються випадки, з якими найчастіше зустрічається кінцевий користувач.

Додатки, що використовують алгоритми машинного навчанная мають переваги над конкретними реалізаціями через те, що подібні розробки можуть у ході навчання виконувати не лише мінімально поставлені задачі, а і типові, подібні їм завдання [1, 2]. Тобто, вчаться на конкретному прикладі та використовують набуті “знання” для вирішення подібних проблем.

Через бурхливий розвиток галузі сьогодні немає якогось єдиного правильного рішення чи стандарту щодо інструментів та технологій для впровадження інтелектуальних систем. Тому вміння оцінювати доцільність та обгрунтованість використання інтелектуальної системи заради отримання результату, а не тільки для наявності самої системи, согодні досить актуальне.

Метою статті є аналіз автоматизації окремих етапів розробки програмного забезпечення з використанням машинного навчання та нейромереж.

Аналіз автоматизації етапів розробки програмного забезпечення з використанням інтелектуальних систем

Як відомо, основними етапами розробки програмного забезпечення є [3]:

- аналіз вимог до проекту;
- проектування;
- реалізація;
- тестування продукту;
- впровадження та підтримка.

Проведемо аналіз етапів розробки ПЗ, у яких можна використати інструменти машинного навчання. Не можна переоцінити важливість та необхідність такого етапу розробки програмного забезпечення як тестування. Із розвитком технологій тестування стає дедалі більш автоматизованих процесом. У сучасному світі тестовані системи стають дедалі складнішими, а тому і потребують нових підходів до процесу.

Тестування систем, які не завжди повертають однакові відповіді, вимагає нових підходів. Це особливо актуально при тестуванні систем, відповіді яких адаптуються до того, що вони дізналися з попередніх ітерацій.

Тестування програмного забезпечення, теоретично, є досить простою діяльністю. Для кожного вводу має бути визначений і відомий результат. Ми вводимо значення, робимо вибір, або переміщаємось у програмі та порівнюємо фактичний результат з очікуваним. Якщо вони збігаються, ми погоджуємося і рухаємось далі. Якщо вони цього не роблять, можливо, ми маємо помилку.

Справа в тому, ми вже знаємо, яким має бути результат. Але іноді результат не є чітко визначеним, існує деяка невизначеність, і отримуємо розбіжності в тому, чи конкретний результат є помилкою.

Але існує такий тип програмного забезпечення, де наявність визначеного виходу вже не є правильним. Власне, два типи. Одним з них є системи машинного навчання; інший - предиктивна аналітика.

Існує різниця між цими двома. Більшість систем машинного навчання базуються на нейронних мережах. Нейронна мережа - це набір багатошарових алгоритмів, змінні яких можна регулювати за допомогою процесу навчання. Процес навчання передбачає використання відомих вхідних даних для створення результатів, які потім порівнюються з відомими результатами. Коли алгоритми відповідають відомим результатам з бажаним ступенем точності, генерується остаточний код.

Сьогодні це і є тим, що ми називаємо штучним інтелектом.

На відміну від них, предиктивна аналітика вносить корективи до алгоритмів у виробництві, виходячи з результатів, що надходять у програмне забезпечення. Іншими словами, програма краще розуміє, як застосовувати свої правила на основі того, як ці правила працювали в минулому. Ці системи продовжують адаптуватися після їх впровадження.

Обидва типи систем мають спільні речі. З одного боку, жодна не отримує "точний" результат. Фактично, іноді вони можуть видати очевидно невірний результат. Але вони надзвичайно корисні в ряді ситуацій, коли вже існують

дані про взаємозв'язок між записаними вхідними матеріалами та передбаченими результатами.

Прикладом можна привести створення тестової програми для раніше розробленої гри. Замість того, щоб тестувати потенційно відомі та можливі помилки, програма просто буде вчитися грати у гру – те, власне для чого ця гра і була створена. І так у ході подальшого навчання будуть виявлені саме ті помилки, з якими з великою вірогідністю може зустритися кінцевий користувач. Завдяки цьому їх можна помітити та виправити ще на стадії розробки продукту.

У проекті кожний об'єкт має власну нейронну мережу, що використовується як ІІ-мозок для проходження ігри. Вона складається з наступних трьох шарів:

1. шар вхідних даних з двома нейронами: горизонтальна відстань до найближчого проміжку та різниця висот (1)
2. внутрішній шар
3. шар вихідних даних, що створює дію: зробити зліт або ж не робити нічого

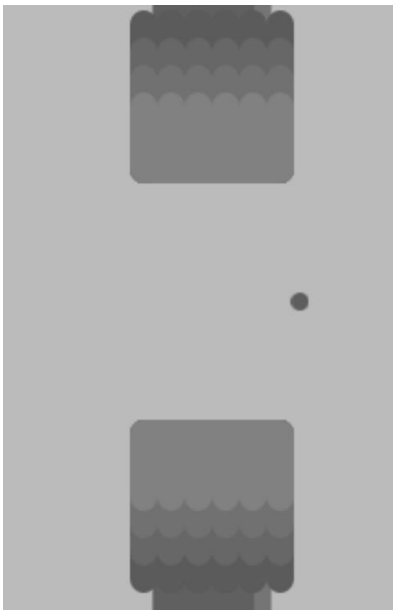

Рис. 1. Очікуваний ідеальний проміжок прольоту

Також було використано генетичный алгоритм. Ось основні етапи реалізації нашого генетичного алгоритму:

1. створюємо популяцію з 10 птахів випадково
2. дозволяємо їм усім грати одночасно
3. у ході цього визначимо коефіцієнт адаптації – те, наскільки птах зробив задачу краще інших
4. після смерті всіх об'єктів оцінюємо результати та обираємо найкращих
5. повторюємо етап 2

В результаті декількох ітерацій ми можемо отримати майже ідеального гравця. В табл. 1 наведено результати адаптації різних поколінь, які виражено у відносних величинах. Для досягнення цієї мети ми використовували два підходи до алгоритмів машинного навчання: штучні нейронні мережі та генетичний алгоритм.

Таблиця 1

Таблиця зведених результатів адаптації різних поколінь

	Покоління 6	Покоління 7
Значення адаптації	562.67	2696.287
	561.53	2831.27
	557.29	446.57

І тестові практики, і результати повинні змінюватися, щоб тестувати програми, які не ведуть себе так само, як традиційне програмне забезпечення. У разі роботи із над машинним навчанням та інтелектуальними програмами, приведені підходи є хорошим початком у цьому напрямку [4, 5].

Ще одним із найважливіших етапів розробки ПЗ є етап реалізації. Проведемо аналіз шляхів застосування інтелектуальних систем для автоматизації даного етапу.

Для такого етапу розробки програмного забезпечення, як реалізація, необхідне залучення великої кількості інструментів та технологій. Першим та

найпростішим прикладом, де було б корисним застосування інтелектуальних систем, можна назвати сам процес написання коду продукту. А саме інструментів, що для цього застосовуються: редакторів. Існує велика кількість різновидів редакторів та інтегрованих середовищ розробки. Але все ж таки основою їх ціллю є прискорення, полегшення та систематизація процессу написання коду. І тут за допомогою впровадження інтелектуальних систем у такі інструменти, можна значно підвищити їх ефективність. Так, наприклад, можна розширити можливості автодоповнення середовища розробки. Зазвичай, це робиться на основі статистичних даних: як часто ті чи інші опції вибрано, деякі стандартні параметри, пропозиції щодо автоматичного закінчення інструкцій.

Але з використанням інтелектуальних систем можна вивести ці інструменти на суттєво новий рівень: у ході роботи середовище розробки буде постійно навчатися, збирати дані про дії користувача і на їх основі “підлаштовуватися” під нього або цілу групу розробників. Тут чітко прослідкується наявність предиктивної аналітичної системи, яка також була описана з точки зору застосування у галузі тестування.

Сьогодні традиційним методом створення програмного забезпечення є слідування вимогам, що висунуто до нього, і відповідна їх реалізація з використанням різних технологій. На виході матимемо продукт, який буде виконувати поставлені перед ним задачі. Але із використанням інтелектуальних систем та автоматизації сьогодні можно виділити інший підхід: створення системи не із чіткими інструкціями щодо виконання задач в залежності від вхідних даних, а системи, яка б мала можливість навчится реалізовувати вхідні задачі після певних етапів її навчання.

Наприклад, замість того, щоб реалізовувати алгоритми перетворення графічних зображень, фотокорекції або фільтрів, створити систему, яка б мала деякі вхідні дані і завдяки цьому закрито навчалася виконувати подібні задачі. В якості прикладу, первісне зображення та необхідний результат перетворення зображені на рис. 2.

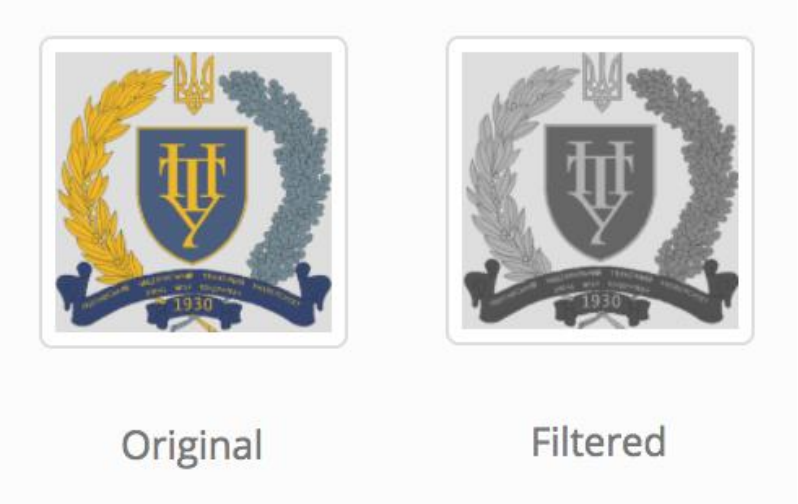

Рис. 2. Вхідне та вихідне зображення

Тобто, замість створення системи для перетворення вхідного зображення на чорно-біле, система приймала б вхідне зображення та вихідне чорно-біле (або будь-яке інше) і вже далі могла застосувати отримані дані для перетворення будь-якого вхідного зображення на необхідне. Тобто на основі двох зображень створюється алгоритм для перетворення подальших даних на чорно-білі зображення. Аналогічно можна замінити чорно-билій фільтр на сепію чи будь-що інше.

Для реалізації перетворення використана модель перцептрону, що показано на рис. 3 [6].

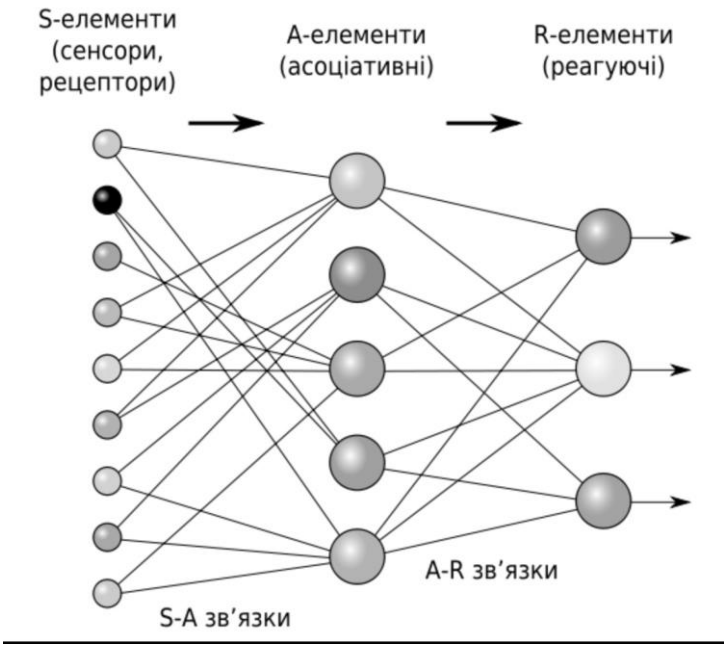

Рис. 3. Логічна схема перцептрону з трьома виходами

Перцептрон складається з трьох типів елементів, а саме: сигнали, що надходять від датчиків, передаються до асоціативних елементів, а відтак до реагуючих. Таким чином, перцептрони дозволяють створити набір «асоціацій» між вхідними стимулами та необхідною реакцією на виході.

Висновок

У роботі проведено аналіз автоматизації етапів розробки програмного забезпечення з використанням інтелектуальних систем. Було проаналізовано використання інтелектуальних систем під час етапу тестування. Також було розглянуто шляхи автоматизації етапу реалізації додатків – не розробка продукту із чітко поставленими цілями, а натомість створення системи, яка б у ході навчання могла втілювати поставлені задачі.

В подальшому, дослідження будуть спрямовані на реалізацію програмного забезпечення з використанням інтелектуальних систем.

Посилання

1. *Machine-learning revolutionizes software development [електронний ресурс]. – Режим доступу: https://www.sciencedaily.com/releases/2010/04/100420161222*
2. *How can machine learning improve software development [електронний ресурс]. – Режим доступу: https://www.quora.com/How-can-machine-learning-improve-software-development*
3. *Software development process [електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/Software_development_process*
4. *Machine learning impacting software testing [електронний ресурс]. – Режим доступу: https://www.sogeti.ie/explore/sogeti-ireland-blog/machine-learning-impacting-software-testing/*
5. *Testing software in the age of machine learning [електронний ресурс]. – Режим доступу: https://techbeacon.com/moving-targets-testing-software-age-machine-learning*
6. *Perceptron [електронний ресурс]. – Режим доступу: https://es.wikipedia.org/wiki/Perceptr%C3%B3n*

Authors: Vorona Oleksandr, Vasyliev Kostiantyn

AUTOMATION ANALYSIS OF SOFTWARE DEVELOPMENT STAGES

Abstract. An analysis of automation of software development stages with using intelligent systems. The use of intelligent systems in the stages of testing and applications implementing was analysed.

Keywords: testing, software, predictive analytics, machine learning.

Авторы: Ворона А.О., Васильев К.А.

АНАЛИЗ АВТОМАТИЗАЦИИ ОТДЕЛЬНЫХ ЭТАПОВ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Аннотация. В работе проведен анализ автоматизации этапов разработки программного обеспечения с использованием интеллектуальных систем. Было проанализировано использование интеллектуальных систем в этапах тестирования и реализации приложений.

Ключевые слова: тестирование, программное обеспечение, предиктивная аналитика, машинное обучение.