ДОДАТОК А

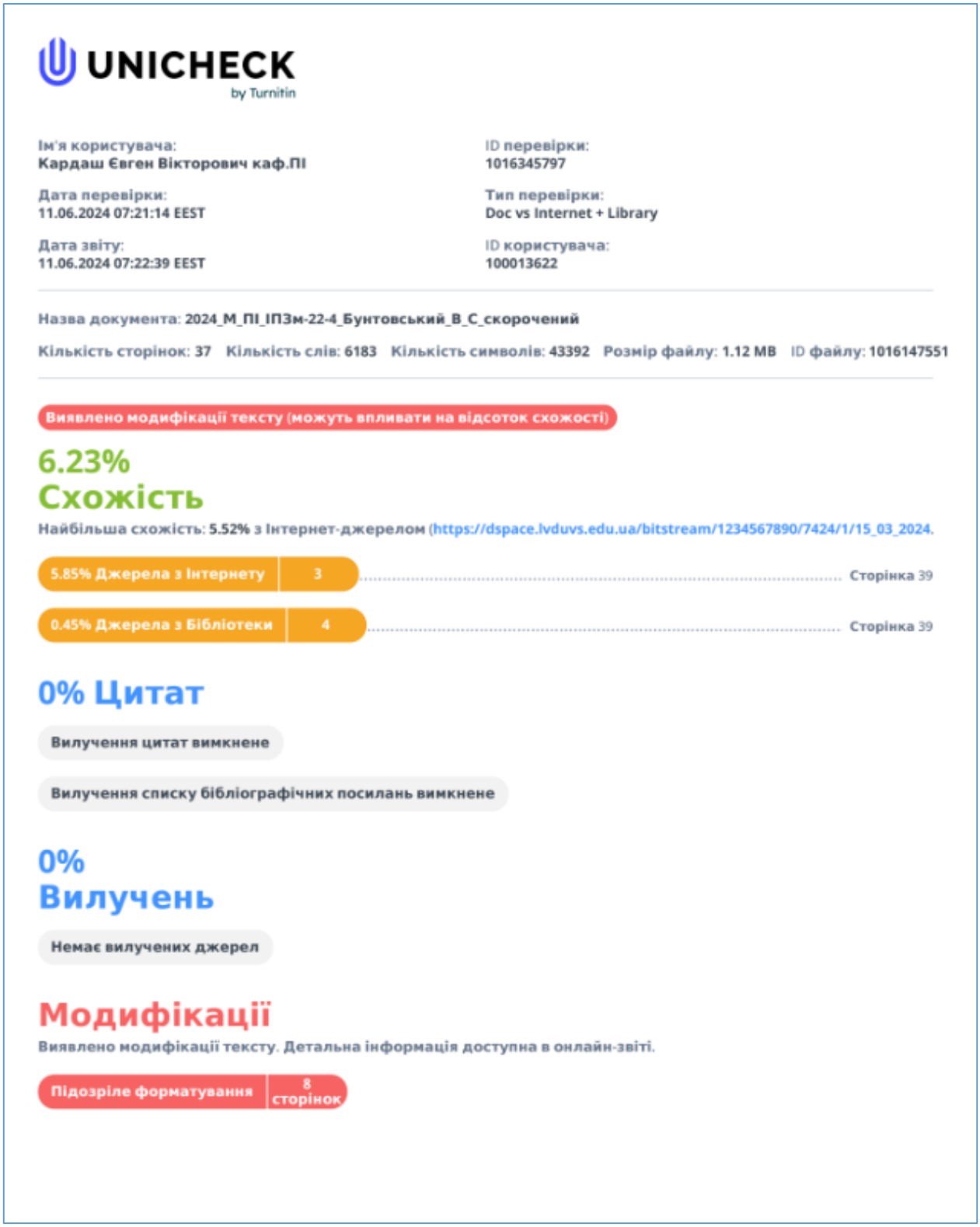
Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії

1. Каук В. І. Генеративний штучний інтелект – креативний помічник дизайнера / В. І. Каук // Поліграфічні, мультимедійні та web-технології. Сучасний стан: монографія. – Харків: ТОВ «Друкарня Мадрид», 2023. – С. 283-294.

2. Турута О. П. Використання моделі dall-e для створення у програмній системі на основі штучного інтелекту для поштового клієнту GMAIL / О. П. Турута, Ж. В. Дейнеко, І. Є. Мічурін // Поліграфічні, мультимедійні та web-технології : тези доп. IX Міжнар. наук.-техн. конф., 14-18 травня 2024 р. – Т. 1. – Харків: ТОВ «Друкарня Мадрид», 2024. – С. 95-96.

ДОДАТОК Б

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016345797

Дата перевірки:
11.06.2024 07:21:14 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2024 07:22:39 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗм-22-4_Бунтовський_В_С_скорочений

Кількість сторінок: 37 Кількість слів: 6183 Кількість символів: 43392 Розмір файлу: 1.12 MB ID файлу: 1016147551

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.23%
Схожість

Найбільша схожість: 5.52% з Інтернет-джерелом (https://dspace.lvduvs.edu.ua/bitstream/1234567890/7424/1/15_03_2024.

5.85% Джерела з Інтернету 3 ..... Сторінка 39

0.45% Джерела з Бібліотеки 4 ..... Сторінка 39

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

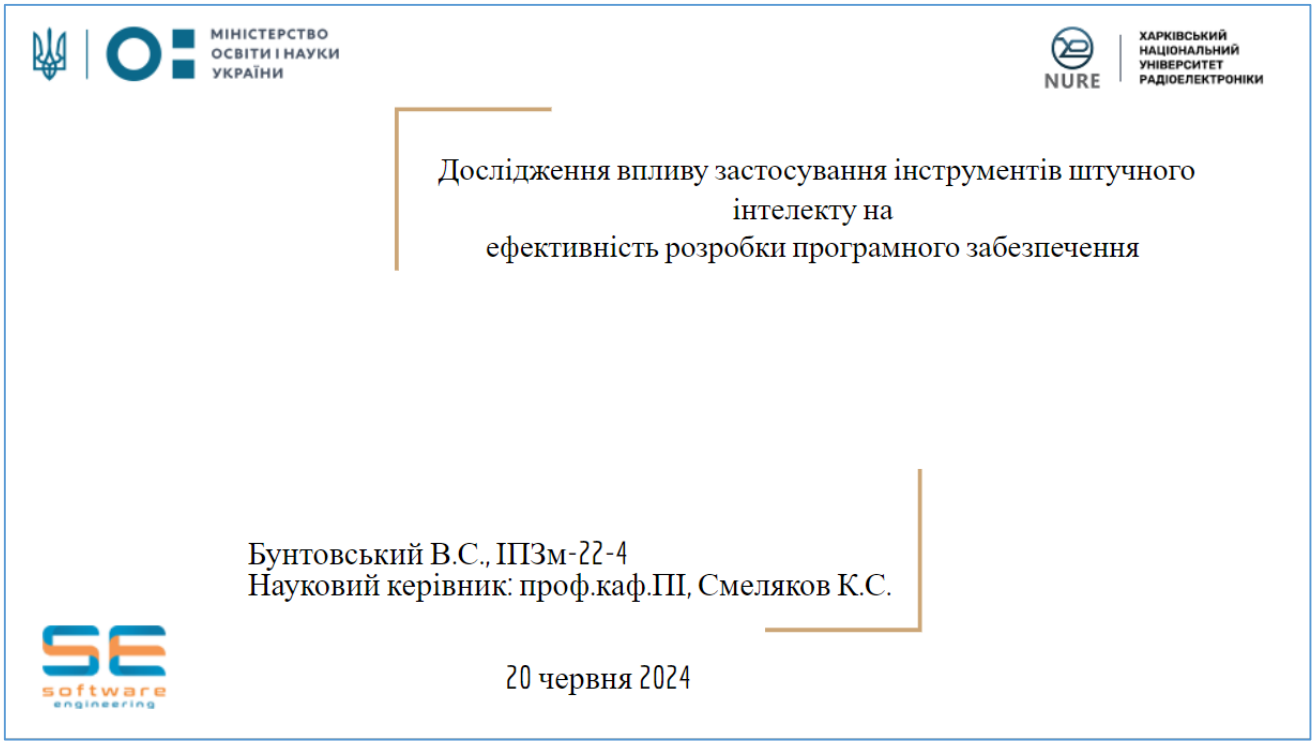
Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 8 сторінок

ДОДАТОК В

Слайди презентації



Огляд літератури (аналогів)

Ключові джерела:

- Haykin S. Neural Networks. A comprehensive Foundation / S. Haykin // Prentice Hall, Inc. N.J. - 2ed. - 1999. -P. 690.
- "A Survey on the Use of Machine Learning for Software Security" by M. Shahbaz et al. (IEEE Transactions on Reliability, 2019)
- "Software defect prediction using supervised learning and ensemble techniques" by Y. Ma et al. (Journal of Systems and Software, 2020)
- "Using artificial intelligence to improve real time decision-making in code review" by Carlos Gómez Teshima (ACM Computing Surveys, 2021)

Постановка задачі

Проблема, яку досліджує дана дипломна робота, полягає у визначенні ступеня впливу інструментів штучного інтелекту на ефективність процесів розробки програмного забезпечення.

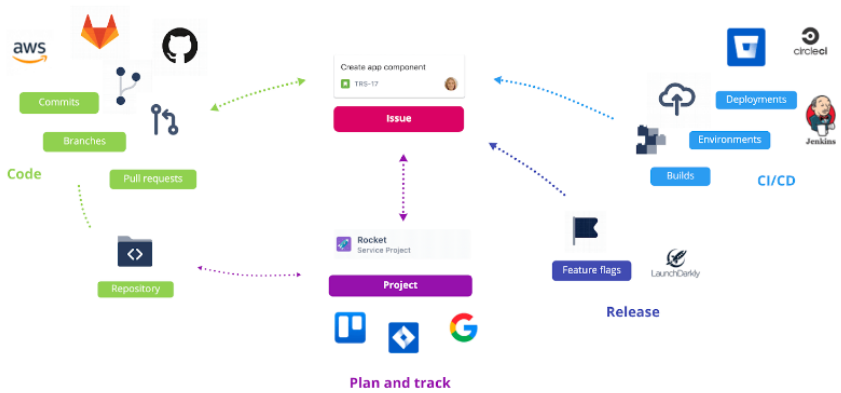
Очікуваний результат:

- Визначені конкретні області програмної інженерії, де застосування ШІ є найбільш ефективним, та ідентифіковані типи задач, для яких ШІ може надати значні переваги.
- Оцінений вплив ШІ на продуктивність розробників за допомогою кількісних і якісних метрик, таких як час розробки, кількість помилок, та частота внесення змін у код
- Розроблені рекомендації щодо інтеграції інструментів ШІ у процеси розробки, засновані на аналізі переваг і можливих ризиків

Методологія

- Використання **Jira** для трекінгу часу: Для збору даних про час, витрачений на різні аспекти розробки, включаючи розробку нових функцій, виправлення помилок та виконання завдань. Це дозволило аналізувати ефективність робочих процесів і виявити зони для потенційного покращення за допомогою автоматизації.
- Проведення **код ревью** для оцінки якості коду: Код ревью допомогло визначити якість коду, виявити помилки та виробити рекомендації щодо його оптимізації. Цей процес також включав оцінку того, як використання інструментів ШІ впливає на стандарти кодування та мінімізацію помилок.
- Використання статичного аналізатора коду: **Статичний аналіз** дозволяє автоматично виявляти вразливості та помилки в коді, що сприяло підвищенню якості коду та зменшенню часу, необхідного для ручного перегляду коду.

Архітектура система для проведення експериментального дослідження



Опис програмного забезпечення, що було використано у дослідженні

Зміст проведеного експерименту

Вхідні дані:

- 9 розробників (по 3 Java, Front-End, Mobile)
- по 10 тікетів “без AI” та “з AI” для кожного
- Code-review та використання статичного аналізатора
- логування часу розробки

Критерії:

- використаний час
- кількість коментарів з Code-review
- кількість помилок від статичного аналізатора

Direction	Task description	Without AI	With AI
Java	Subscription service	ACH-01	ACH-11
	Defining subscription service entities	ACH-02	ACH-12
	Create REST API endpoint to create new subscription	ACH-03	ACH-13
	Create REST API endpoint to get user`s subscription details	ACH-04	ACH-14
	Create REST API endpoint to update subscription	ACH-05	ACH-15
	Create REST API endpoint to cancel subscription	ACH-06	ACH-16
	Create integration tests for subscription service	ACH-07	ACH-17
	Setup MySQL database schema	ACH-08	ACH-18
	Setup Spring security with JWT	ACH-09	ACH-19
	Create CRUD operations for products	ACH-10	ACH-20

Direction	Task description	Without AI	With AI
Frontend	Login	ACH-21	ACH-31
	Profile	ACH-22	ACH-32
	Add post	ACH-23	ACH-33
	Registration	ACH-24	ACH-34
	Working with followers	ACH-25	ACH-35
	User single page	ACH-26	ACH-36
	Specific group list	ACH-27	ACH-37
	Settings page	ACH-28	ACH-38
	Feedback	ACH-29	ACH-39
	Profile deleting	ACH-30	ACH-40

Результати експерименту

В ході експерименту отримали такі результати - швидкість виконання завдання зросла на **30%**. Кількість зауважень до коду зросла в середньому на **22%**.

Developer	AI	Ticket	Original estimate	Time spent	Comments count	PMD errors count
MobDev_1	Without AI	ACH-41	4	5	1	2
		ACH-42	16	15	6	5
		ACH-43	8	8	4	3
		ACH-44	8	7	2	2
		ACH-45	4	4	2	2
		ACH-46	6	5	1	1
		ACH-47	8	7	4	5
		ACH-48	6	6	2	3
		ACH-49	6	7	3	2
		ACH-50	2	2	0	1

Developer	AI	Ticket	Original estimate	Time spent	Comments count	PMD errors count
MobDev_1	With AI	ACH-51	4	3	2	3
		ACH-52	16	10	4	4
		ACH-53	8	5	3	4
		ACH-54	8	4	3	3
		ACH-55	4	3	3	4
		ACH-56	6	5	2	1
		ACH-57	8	3	4	4
		ACH-58	6	4	2	5
		ACH-59	6	4	4	1
		ACH-60	2	1	2	1

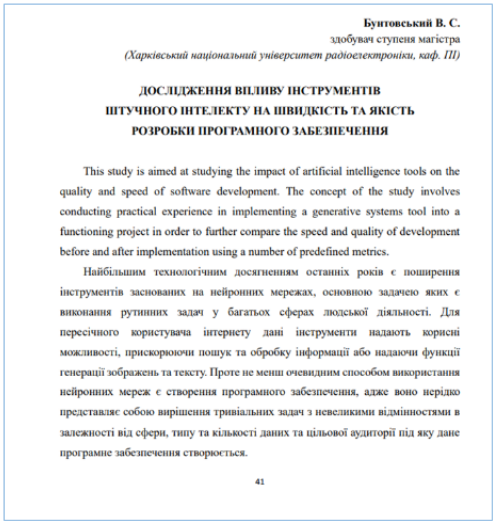
Аналіз отриманих результатів

В ході експерименту було визначено що використання інструментів генерації програмного коду збільшує продуктивність до 30%. (40% у випадку якщо розробник компетентен у роботі з ШІ). Ці показники відповідають даним які можна знайти у відкритих джерелах. Важливим зауваженням є факт того що показники є відмінними в залежності від напрямку розробки.

Direction		Productivity	Quality (comment per MR)
Java	Without AI	1.18*estimated	3.24
	With AI	0.79*estimated	4.26
Frontend	Without AI	1.07*estimated	2.17
	With AI	0.71*estimated	3.85
Mobile	Without AI	0.96*estimated	2.9
	With AI	0.68*estimated	3.51

Публікація результатів

Роботу було представлено на Всеукраїнському круглому столі “**Штучний інтелект у правовій практиці: межі та можливості**” 15 березня 2024р.



Бунтовський В. С.
здобувач ступеня магістра
(Харківський національний університет радіоелектроніки, каф. ПІ)

ДОСЛІДЖЕННЯ ВПЛИВУ ІНСТРУМЕНТІВ ШТУЧНОГО ІНТЕЛЕКТУ НА ШВИДКІСТЬ ТА ЯКІСТЬ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

This study is aimed at studying the impact of artificial intelligence tools on the quality and speed of software development. The concept of the study involves conducting practical experience in implementing a generative systems tool into a functioning project in order to further compare the speed and quality of development before and after implementation using a number of predefined metrics.

Найбільшим технологічним досягненням останніх років є поширення інструментів заснованих на нейронних мережах, основною задачею яких є виконання рутинних задач у багатьох сферах людської діяльності. Для пересічного користувача інтернету дані інструменти надають корисні можливості, прискорюючи пошук та обробку інформації або надаючи функції генерації зображень та тексту. Проте не менш очевидним способом використання нейронних мереж є створення програмного забезпечення, адже воно нерідко представляє собою вирішення тривіальних задач з невеликими відмінностями в залежності від сфери, типу та кількості даних та цільової аудиторії під яку дане програмне забезпечення створюється.

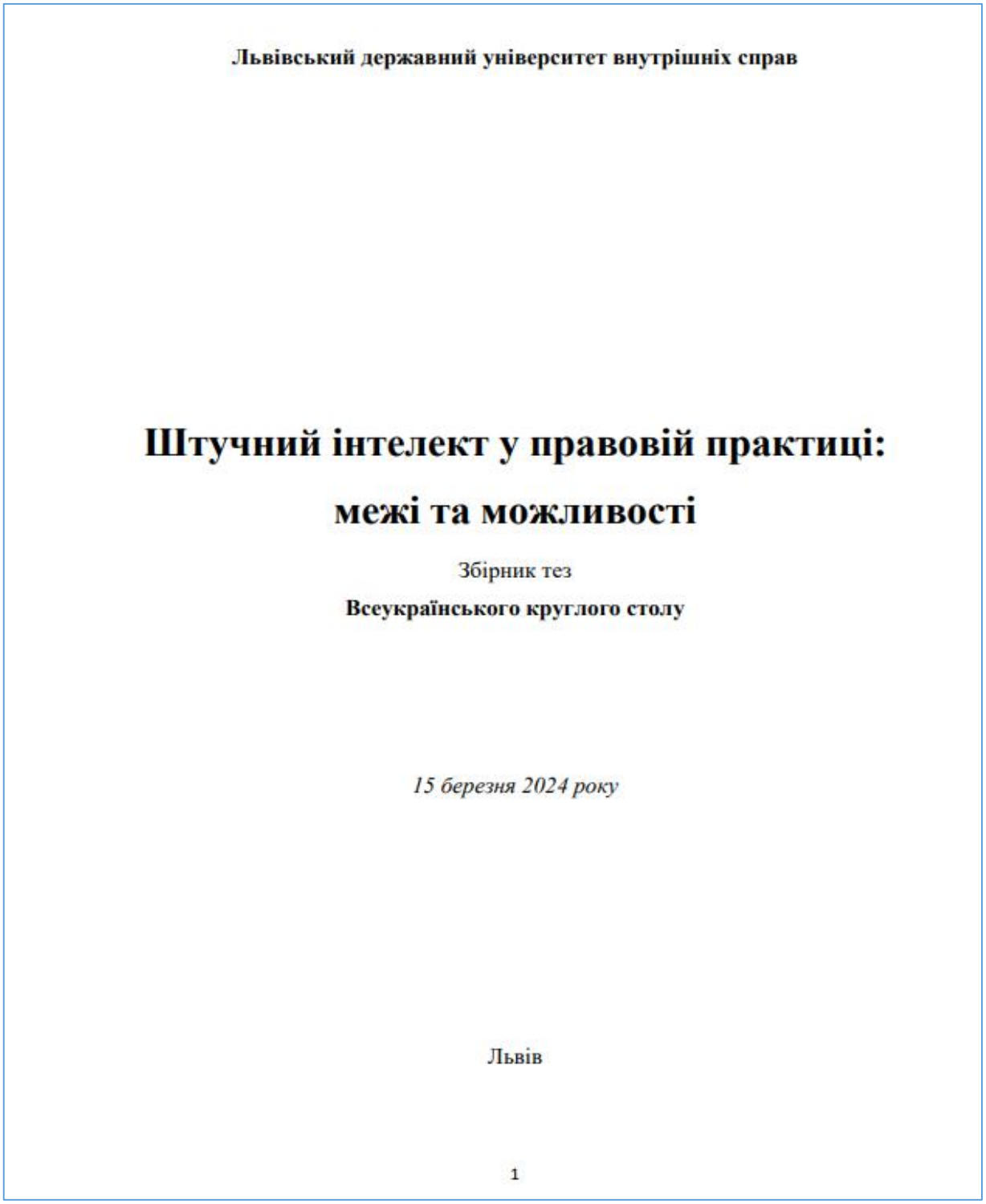
Підсумки

Отримані результати можуть допомогти компаніям які спеціалізуються на розробці програмного забезпечення прийняти рішення щодо впровадження інструментів ШІ в процес розробки на рівні методології.

Для більшості випадків відповідь очевидна - впровадження інструментів ШІ може підвищити швидкість розробки на 30-40% з урахуванням додаткового часу на посилення контролю за якістю коду.

ДОДАТОК Г

Тези доповіді для конференції



Львівський державний університет внутрішніх справ

Штучний інтелект у правовій практиці:
межі та можливості

Збірник тез
Всеукраїнського круглого столу

15 березня 2024 року

Львів

1

Рисунок В.1 – Обкладинка збірника

Бунтовський В. С.
здобувач ступеня магістра
(Харківський національний університет радіоелектроніки, каф. ПІ)

ДОСЛІДЖЕННЯ ВПЛИВУ ІНСТРУМЕНТІВ ШТУЧНОГО ІНТЕЛЕКТУ НА ШВИДКІСТЬ ТА ЯКІСТЬ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

This study is aimed at studying the impact of artificial intelligence tools on the quality and speed of software development. The concept of the study involves conducting practical experience in implementing a generative systems tool into a functioning project in order to further compare the speed and quality of development before and after implementation using a number of predefined metrics.

Найбільшим технологічним досягненням останніх років є поширення інструментів заснованих на нейронних мережах, основною задачею яких є виконання рутинних задач у багатьох сферах людської діяльності. Для пересічного користувача інтернету дані інструменти надають корисні можливості, прискорюючи пошук та обробку інформації або надаючи функції генерації зображень та тексту. Проте не менш очевидним способом використання нейронних мереж є створення програмного забезпечення, адже воно нерідко представляє собою вирішення тривіальних задач з невеликими відмінностями в залежності від сфери, типу та кількості даних та цільової аудиторії під яку дане програмне забезпечення створюється.

Рисунок В.2 – Матеріал власних тез (перша сторінка)

Таким чином, об'єкт дослідження – використання штучного інтелекту (AI) асистентів, зокрема моделей на кшталт GPT (Generative Pre-trained Transformer) та GitHub Copilot, у процесі розробки програмного забезпечення. В цьому контексті, особлива увага приділяється аналізу впливу цих технологій на швидкість та якість процесів розробки. Мета дослідження – оцінка впливу AI асистентів на процеси розробки програмного забезпечення, що включає в себе вимірювання змін у продуктивності та якості коду, а також аналіз ефективності цих інструментів у різних аспектах програмування, від автоматизації рутинних завдань до підтримки складних процесів розробки.

Отже в ході дослідження потрібно знайти підтвердження або спростування для даних припущень:

– Використання AI-асистента пришвидшує роботу програміста;

– Використання AI-асистента знижує якість коду програміста;

– Час на перевірку коду від AI-асистента збільшує час роботи програміста та зменшує продуктивність [1];

– Зростання ефективності різне в залежності від напрямку відділу та науковості проекту;

– Вартість ліцензії на AI-асистента не варта сумарного підвищення ефективності в компанії [1];

– Вартість ліцензії на AI-асистента дає прибуток деяких відділів або проектів.

Для проведення експерименту та перевірки гіпотез треба порівняти роботу програмістів з AI-асистентом та без AI-асистента, працюючих в однаковому середовищі. Треба прийняти до уваги що:

– На вивчення роботи AI-асистента потрібен час тож людина не може давати адекватний результат відразу після початку використання - треба дати якийсь час після якого слід проводити заміри.

– Знаючі що людина приймає участь в експерименті може створити "bias" в даних які ми збираємо.

Рисунок В.3 – Матеріал власних тез (друга сторінка)

– Не можна порівнювати різні відділи або проекти. В ідеалі треба порівнювати результати ефективності тої самої людини до AI-асистента та наприклад через тиждень після початку використання.

– Перевіряти швидкість роботи можна там де використовується якась система контролю розробки.

– Перевіряти якість можна там де використовуються практика «code reviеw» - та де результати такого аналізу можна відстежувати.

Пропонується знайти групу добровольців на проекти що вже довго в роботі та де виконуються усі вищезазначені умови. Для порівняння результатів до та після впровадження AI-асистента можна використати:

– A/B тести

– Trial тести

Також необхідно підрахувати метрики часу / якості роботи групи, впровадити для групи AI-асистент (можуть бути різні асистенти для різних груп для порівняння систем між собою), дати час на освоєння системи та поставити задачу почати використовувати асистента у робочих задачах. Через певний час заміряти метрики по групі знов (час / якість) та визначити наявність приросту швидкості [2] та/або якості та чи покриває він вартість ліцензії.

Приймемо до уваги й потенційні ризики:

– Некоректна або неоптимальна генерація коду, що може призвести до помилок та недоліків у програмному забезпеченні.

– Залежність від сторонніх сервісів та можливість відключення або недоступності цих сервісів.

– Вплив на процеси та ефективність роботи команди розробників, особливо на етапі впровадження системи.

Для перевірки гіпотез про покращення швидкості та якості роботи програмістів можна виконати наступні кроки:

– Впровадження системи у випробувальному режимі для обмеженої групи програмістів.

Рисунок В.4 – Матеріал власних тез (третя сторінка)

– Зібрати дані про час, необхідний для виконання завдань, та якість коду перед і після впровадження системи.

– Порівняти отримані результати з даними до впровадження системи.

– Провести опитування та зібрати зворотній зв'язок від програмістів щодо впливу системи на їх роботу.

– Аналізувати зібрані дані та зворотній зв'язок для визначення покращень і можливих обмежень системи.

Впровадження системи допомоги програмістам в компанії розробника програмного забезпечення має потенціал покращити швидкість та якість роботи програмістів. Порівняння різних варіантів, таких як GitHub Copilot, Amazon CodeWhisperer та StarCoder, допоможе визначити оптимальний варіант для компанії. Потрібно врахувати плюси та мінуси кожної системи, потенційні ризики, спосіб перевірки гіпотез та підрахунок ROI. Належить збалансувати ефективність, вартість та інші фактори для прийняття інформованого рішення щодо впровадження системи допомоги програмістам у компанії.

Список використаних джерел:

1. A Clear Explanation of TransformerNeural Networks - [Електронний ресурс]. - Режим доступу до ресурсу: https://www.linkedin.com/pulse/clear-explanation-transformer-neural-networks-ebin-babu-thomas (дата звернення: 01.03.2024)

2. Zhao H. Global asymptotic stability of Hopfield neural network involving distributed delays / H. Zhao // Neural Networks. - 2004. - Vol. 17, N 1. - P. 48 - 53.

Рисунок В.5 – Матеріал власних тез (четверта сторінка)

ДОДАТОК Д

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-4 (група)

Бунтовський Владислав Сергійович

(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні позначки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

12.06.2024