ВІСНИК ПРИАЗОВСЬКОГО ДЕРЖАВНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
2026р. Серія: Технічні науки Вип. 53. Т. 1
p-ISSN: 2225-6733; e-ISSN: 2519-271X

УДК 004.421.2:004.056:004.738.5 DOI: 10.31498/2225-6733.53.1.2026.359775

МЕТОД ВИЗНАЧЕННЯ ЙМОВІРНОГО ПОХОДЖЕННЯ ТА ЛІЦЕНЗІЙНИХ УМОВ ПРОГРАМНОГО КОДУ, ЗГЕНЕРОВАНОГО ВЕЛИКИМИ МОВНИМИ МОДЕЛЯМИ

Романенко С.О. *ст. викладач, Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, м. Київ, ORCID: https://orcid.org/0009-0004-0240-0777, e-mail: serhi.romanenko@viti.edu.ua*

Активне впровадження великих мовних моделей у процес розроблення програмного забезпечення істотно змінює підходи до створення програмного коду. Одним із найбільш поширених сценаріїв використання таких моделей є автоматична генерація програмних фрагментів, що дозволяє підвищити продуктивність розробників та скоротити час реалізації програмних проєктів. Водночас застосування автоматично згенерованого коду породжує низку нових проблем, пов'язаних із відсутністю інформації про його походження, що ускладнює перевірку надійності джерела, а також може призводити до порушення авторських прав і ліцензійних умов. У роботі розглянуто проблему визначення походження програмного коду, створеного за допомогою великих мовних моделей, та запропоновано підхід до її вирішення. Метою дослідження є розроблення методу встановлення ймовірних джерел програмного коду та визначення можливих ліцензій його використання на основі аналізу подібності між згенерованими та знайденими у відкритих джерелах фрагментами. Запропонований підхід передбачає поєднання можливостей мовних моделей із механізмами вебпошуку, подальшим аналізом текстової подібності та виявленням клонів програмного коду. У результаті формується перелік вебресурсів, що можуть містити схожі фрагменти коду, а також здійснюється спроба автоматичного визначення їхніх ліцензійних умов. Отримані результати демонструють можливість ефективної фільтрації нерелевантних джерел та підвищення прозорості використання автоматично згенерованого програмного забезпечення. Наукова новизна дослідження полягає у поєднанні методів аналізу подібності програмного коду з інструментами вебпошуку для встановлення його ймовірного походження. Практична значущість роботи полягає у можливості використання запропонованого підходу для підвищення надійності програмних продуктів і зменшення ризиків порушення ліцензійних вимог. Перспективним напрямом подальших досліджень є удосконалення алгоритмів пошуку програмного коду та розширення методів автоматичного визначення його ліцензійних характеристик.

Ключові слова: *великі мовні моделі; генерація програмного коду; походження програмного коду; аналіз подібності коду; ліцензування програмного забезпечення; програмна інженерія; штучний інтелект.*

Постановка проблеми

Стрімкий розвиток технологій штучного інтелекту призвів до широкого впровадження великих мовних моделей (LLM) у процес розроблення програмного забезпечення. Сучасні системи на основі LLM здатні генерувати програмний код різними мовами програмування, допомагати у написанні алгоритмів, створенні тестів та документуванні програмних систем. Завдяки цьому значно підвищується продуктивність розробників та скорочується час створення програмних продуктів.

Разом із тим використання автоматично згенерованого програмного коду супроводжується рядом проблем. Однією з ключових є відсутність достовірної інформації про походження такого коду. У традиційній практиці повторного використання програмних компонентів із відкритих джерел існує можливість встановити автора або джерело коду, що дозволяє оцінити його надійність та перевірити ліцензійні умови використання. У випадку ж із кодом, створеним мовними моделями, така інформація часто відсутня.

Додаткову складність становить явище часткового відтворення фрагментів коду, що містилися у навчальних наборах даних LLM. У результаті цього згенерований код може бути подібним або навіть ідентичним до вже існуючих реалізацій, які поширюються під певними ліцензіями. Це створює ризик порушення авторських прав або ліцензійних вимог під час використання такого коду у програмних проєктах.

Таким чином, виникає наукова та практична проблема розроблення методів визначення можливого походження програмного коду, згенерованого LLM, а також автоматичного встановлення ліцензійних умов його використання.

Аналіз останніх досліджень та публікацій

Проблематика використання LLM у програмній інженерії активно досліджується в останні роки. Значна кількість наукових робіт присвячена оцінюванню ефективності генерації програмного коду за допомогою LLM, а також аналізу якості та коректності отриманих результатів.

У ряді досліджень розглядається явище memorization, яке полягає у здатності моделей відтворювати фрагменти даних, що використовувалися під час їх навчання. У контексті програмування це означає можливість появи у згенерованому коді фрагментів, які вже існують у відкритих репозиторіях програмного забезпечення [1-6].

Інші дослідження присвячені аналізу впливу параметрів генерації тексту на ймовірність відтворення ліцензованого програмного коду. Зокрема, встановлено, що збільшення контексту у запитах до мовної моделі може підвищувати ймовірність відтворення

фрагментів програмного коду з відкритих джерел, тоді як зміна параметрів генерації може зменшувати цей ризик [7-12].

Окремим напрямом досліджень є аналіз походження програмного коду за допомогою методів виявлення клонів програмних фрагментів та аналізу текстової подібності. Такі підходи дозволяють виявляти схожі фрагменти програмного коду у великих репозиторіях та визначати їх потенційні джерела [13-15].

Разом із тим існуючі дослідження здебільшого орієнтовані на аналіз навчальних наборів даних LLM або на перевірку якості згенерованого коду. Питання автоматичного визначення походження коду та його ліцензійних характеристик залишаються недостатньо дослідженими [8, 16].

Мета статті

Метою статті є розроблення методу визначення ймовірного походження програмного коду, згенерованого LLM, та встановлення можливих ліцензій його використання на основі аналізу подібності між згенерованими та знайденими у відкритих джерелах фрагментами програмного коду.

Матеріали та методи

Об’єктом дослідження є програмний код, згенерований LLM у процесі виконання задач програмування.

Предметом дослідження є методи визначення походження такого коду на основі аналізу подібності з фрагментами програмного забезпечення, що розміщені у відкритих вебресурсах.

Для досягнення поставленої мети використано такі методи:

аналіз текстової подібності програмного коду;

методи виявлення клонів програмного коду;

використання LLM для формування пошукових запитів;

аналіз метаданих вебресурсів та програмних репозиторіїв;

автоматичне визначення ліцензій програмного забезпечення.

Запропонований підхід передбачає використання можливостей LLM для пошуку у мережі Інтернет вебресурсів, що можуть містити фрагменти програмного коду, подібні до згенерованих.

Виклад основного матеріалу

У даній роботі запропоновано підхід, призначений для визначення ймовірного походження програмного коду, згенерованого LLM, а також для встановлення можливих ліцензій його подальшого використання. Запропонований підхід може бути реалізований у вигляді програмного розширення для інтегрованих середовищ розроблення програмного забезпечення, зокрема в середовищі VSCode, із використанням можливостей системи Copilot [5].

Запропоноване рішення передбачає два основні режими функціонування.

1. генерація коду з подальшим пошуком джерел (режим 1).

В даному режимі розробник формує запит до мовної моделі через інтерфейс AICertify для автоматичної генерації фрагмента програмного коду. Отриманий результат використовується як основа для подальшого аналізу: згенерований код автоматично застосовується для виконання вебпошуку з метою виявлення ресурсів мережі Інтернет, які можуть містити подібні або ідентичні фрагменти програмного коду.

2. пошук можливих джерел для наявного фрагмента коду (режим 2).

В даному режимі розробник безпосередньо вибирає певний фрагмент програмного коду в інтегрованому середовищі розроблення та ініціює процедуру пошуку потенційних джерел його походження. Слід зазначити, що такий режим може використовуватися не лише для автоматично згенерованого коду, але й для будь-якого фрагмента програмного забезпечення, походження якого потребує уточнення.

Запропонований підхід ґрунтується на поєднанні методів аналізу текстової подібності та виявлення клонів програмного коду. Це дозволяє визначати можливі джерела походження фрагментів програмного коду, після чого для кожного знайденого ресурсу виконується аналіз метаданих та вмісту вебсторінок з метою встановлення відповідної інформації про ліцензування.

На відміну від підходів, подібних до системи SourceGraph [8], які переважно орієнтовані на перевірку того, чи може згенерований код бути частиною навчальних наборів даних LLM, запропонований підхід передбачає безпосередній пошук відповідних фрагментів програмного коду у відкритих інтернет-джерелах. Такий підхід дозволяє визначати можливі джерела походження програмного коду та пов’язані з ними ліцензійні умови навіть у випадках, коли навчальні дані LLM залишаються недоступними або невідомими.

Було проведено попереднє експериментальне оцінювання запропонованого підходу із використанням 112 завдань програмування з набору даних HumanEval [8] та 108 завдань з набору MBPP [15]. Отримані результати свідчать про перспективність запропонованого підходу. Зокрема, навіть у випадках, коли великі мовні моделі не завжди надають релевантні посилання на джерела програмного коду, застосування механізмів аналізу подібності дозволяє відфільтровувати нерелевантні результати та залишати лише ті посилання, які з високою ймовірністю можуть бути пов’язані з походженням згенерованого коду.

Архітектура

На рис. 1 представлено архітектуру запропонованого підходу. Концепція передбачає інтеграцію з інструментами підтримки розроблення програмного забезпечення, зокрема з системами генерації коду на основі LLM, такими як AICertify, що може

використовуватися в середовищі VSCode. Взаємодія з мовною моделлю здійснюється через спеціалізованих агентів, які виступають точками входу для реалізації різних сценаріїв використання запропонованого підходу.

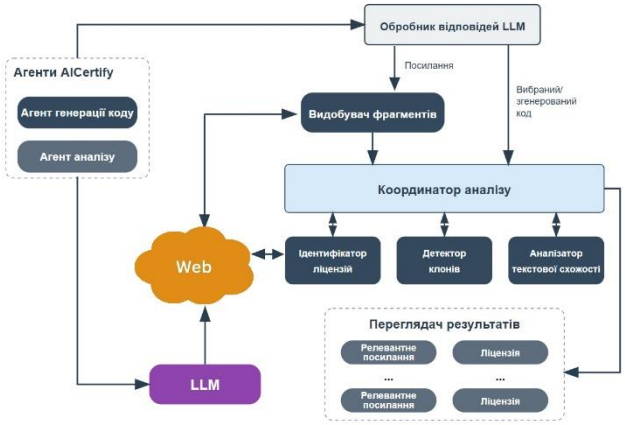

Рис. 1 – Архітектура запропонованого рішення

У межах архітектури передбачено два основні агенти, які відповідають двом режимам функціонування системи: режиму генерації коду з подальшим аналізом джерел та режиму аналізу вже наявного фрагмента коду). Для отримання можливих джерел походження коду використовується мовна модель із підтримкою веб-пошуку. У якості прикладу реалізації може застосовуватися API мовних моделей, що забезпечує доступ до вебпошуку[12], хоча запропонований підхід не обмежується використанням конкретної реалізації.

Для підтримки різних режимів функціонування в архітектурі передбачено два спеціалізовані агенти.

Перший агент використовується у сценарії, коли розробник звертається до мовної моделі із запитом на генерацію програмного коду. У такому випадку модель спочатку генерує фрагмент коду відповідно до поставленого завдання. Після цього отриманий результат використовується для формування додаткового запиту до мовної моделі з метою виконання веб-пошуку та отримання посилань на ресурси, які можуть містити подібні або ідентичні фрагменти програмного коду [2]. Прикладом такого запиту може бути пошук джерел, що містять інформацію або реалізації, подібні до згенерованого фрагмента коду.

Другий агент використовується у випадку, коли розробник має готовий фрагмент програмного коду та бажає визначити його можливе походження. У цьому режимі формується структурований запит, що містить вибраний фрагмент коду та інформацію про мову програмування, після чого виконується пошук веб-ресурсів, які можуть містити подібні реалізації [1].

Модуль відповідей мовної моделі

Компонент оброблення відповідей мовної моделі призначений для перетворення неструктурованого результату роботи LLM – зокрема, згенерованого програмного коду та отриманих вебпосилань – у структуровані дані, які можуть бути використані для подальшого аналізу. Такі дані передаються до модуля аналізу подібності та модуля визначення ліцензій.

Модуль видобування фрагментів коду

Даний компонент аналізує HTML-вміст отриманих вебресурсів з метою видобування фрагментів програмного коду. Оскільки структура вебконтенту різних ресурсів є неоднорідною, застосовуються спеціалізовані стратегії видобування, орієнтовані на конкретні домени. Наприклад, для ресурсів типу GeeksforGeeks програмний код може бути отриманий із вкладених тегів та , тоді як для репозиторіїв GitLab використовується доступ до прямих (raw) версій файлів. Архітектура підходу передбачає можливість розширення списку підтримуваних ресурсів шляхом додавання нових правил видобування.

Модуль координації аналізу коду

Даний компонент об’єднує фрагменти коду, отримані з веб-ресурсів, із вибраним або згенерованим фрагментом коду та виконує процедури визначення подібності між ними. Метою цього етапу є фільтрація нерелевантних посилань і збереження лише тих джерел, які містять програмний код із високим рівнем відповідності.

У межах запропонованого підходу можуть використовуватися два типи методів аналізу:

метрики текстової подібності, що оцінюють схожість фрагментів коду на основі їх текстового представлення;

методи виявлення клонів програмного коду, спрямовані на виявлення структурно подібних реалізацій.

Зокрема, для виявлення клонів може застосовуватися інструмент CCFinderSW [13], який дозволяє ідентифікувати потенційні клони між парами програмних фрагментів. Аналіз текстової подібності може здійснюватися із використанням векторного підходу, відповідно до якого кожен фрагмент коду токенізується, перетворюється на векторне представлення, після чого між векторами обчислюється косинусна міра подібності.

Модуль ідентифікації ліцензії

Одним із функціональних компонентів запропонованого підходу є модуль ідентифікації ліцензій, призначений для визначення умов повторного використання та розповсюдження фрагментів програмного коду, отриманих із відкритих інтернет-джерел.

Для досягнення цієї мети передбачено використання декількох стратегій визначення ліцензій. У випадку, коли отримані посилання вказують на репозиторії GitHub, система може звертатися до відповідного API для отримання метаданих репозиторію, зокрема інформації про ліцензію програмного забезпечення. Якщо така інформація відсутня у метаданих, здійснюється пошук файлу ліцензії в структурі репозиторію (наприклад, файлів типу TERMS, TERMS.txt тощо). У разі виявлення такого файлу його вміст може бути проаналізований за допомогою спеціалізованих інструментів

p-ISSN: 2225-6733; e-ISSN: 2519-271X

класифікації ліцензій, зокрема FOSSology [6], що дозволяє автоматично визначити тип ліцензії.

В інших випадках визначення ліцензії може базуватися на політиці ліцензування відповідного веб-ресурсу. Наприклад, відомо, що всі фрагменти коду, опубліковані на платформі GeeksforGeeks, розповсюджуються відповідно до ліцензії CC BY 4.0. З цією метою в межах запропонованого підходу може підтримуватися база відповідностей типу домен – ліцензія.

Для веб-ресурсів, для яких відсутні явні правила ліцензування, здійснюється аналіз HTML-коду вебсторінки з метою пошуку ключових слів, що відповідають відомим типам ліцензій. Для цього використовується попередньо сформований перелік ліцензій, заснований на стандарті CycloneDX [10]. У разі виявлення відповідних збігів інформація використовується для визначення потенційної ліцензії програмного коду.

Модуль представлення результатів

Після завершення етапів аналізу подібності програмного коду та визначення ліцензійних характеристик виконується процедура фільтрації отриманих результатів. На основі заданих порогових значень подібності зберігаються лише ті пари фрагментів коду, для яких показники подібності перевищують встановлений поріг. Такий підхід дозволяє зменшити кількість нерелевантних результатів та забезпечити відображення лише потенційно значущих джерел.

Отримані результати агрегуються та можуть бути представлені у вигляді структурованої таблиці, що містить перелік URL-адрес можливих джерел походження програмного коду разом із відповідною інформацією про ліцензійні умови його використання. Такий спосіб представлення результатів дозволяє розробникам швидко оцінити можливі джерела походження коду та врахувати відповідні ліцензійні обмеження під час його подальшого використання.

Обмеження запропонованого підходу

Запропонований підхід має певні обмеження, які необхідно враховувати під час його застосування.

По-перше, визначення подібності між фрагментами коду дозволяє виявити лише ймовірні джерела походження, однак не гарантує, що згенерований код безпосередньо походить саме з відповідного веб-ресурсу. Крім того, на поточному етапі підхід не дозволяє однозначно визначити, який саме із порівнюваних фрагментів коду був згенерований мовною моделлю.

По-друге, ефективність запропонованого підходу значною мірою залежить від можливостей механізмів веб-пошуку, що використовуються мовною моделлю. Додатково на точність результатів можуть впливати характеристики застосованих алгоритмів виявлення клонів програмного коду та інструментів автоматичної класифікації ліцензій.

Застосування

Для перевірки працездатності запропонованого підходу може бути реалізовано експериментальний прототип у вигляді програмного розширення для інтегрованого середовища розроблення VSCode. Така реалізація може передбачати використання зовнішніх бібліотек та інструментів, зокрема компонентів для аналізу клонів програмного коду та автоматичної класифікації ліцензій.

У межах запропонованої архітектури підтримуються два основні сценарії застосування системи:

аналіз коду, згенерованого мовною моделлю під час взаємодії з користувачем;

аналіз довільного фрагмента програмного коду, вибраного безпосередньо у середовищі розроблення.

У першому сценарії розробник формує запит до мовної моделі щодо генерації фрагмента програмного коду для виконання певного завдання. Наприклад, таким завданням може бути реалізація алгоритму рекурсивного розпакування архіву у форматі ZIP. Після отримання відповіді мовної моделі згенерований код використовується як основа для подальшого аналізу.

Зокрема, на основі отриманого фрагмента коду формується запит до системи вебпошуку з метою виявлення інтернет-ресурсів, які можуть містити подібні або ідентичні реалізації. Отримані результати проходять подальший етап оброблення, під час якого здійснюється видобування фрагментів програмного коду з відповідних веб-сторінок та їх порівняння із згенерованим кодом.

На основі значень метрик подібності виконується фільтрація отриманих результатів. Це дозволяє усунути нерелевантні посилання, зокрема ті, що стосуються загального опису завдання програмування, але не містять безпосередньо відповідних фрагментів коду. Для кожного знайденого джерела, яке містить подібний програмний код, додатково виконується процедура визначення ліцензії. У результаті розробник отримує інформацію про можливі джерела походження фрагмента коду та пов’язані з ними ліцензійні умови, що може бути корисним під час подальшого використання або розповсюдження програмного забезпечення.

Другий сценарій передбачає аналіз уже наявного фрагмента програмного коду. У даному випадку розробник може вручну вибрати відповідний фрагмент у кодовій базі та ініціювати процедуру визначення його можливого походження.

Після цього система формує пошуковий запит, що містить вибраний фрагмент коду та інформацію про мову програмування. Отримані результати вебпошуку проходять аналогічні етапи оброблення, які включають видобування програмних фрагментів із вебресурсів, аналіз подібності та визначення можливих ліцензійних умов.

Застосування такого підходу дозволяє не лише аналізувати код, згенерований LLM, але й досліджувати походження будь-якого фрагмента програмного забезпечення, що може бути корисним під час аудиту програмних систем, перевірки ліцензійної сумісності або аналізу повторного використання коду.

Наступним етапом буде виклик другого агента, який автоматично створює запит до LLM для

отримання посилань на потенційні джерела, які містять код, подібний до вибраного коду. Відповідь моделі містить список посилань, що вказують на потенційно релевантні вебсторінки. З цього моменту конвеєр продовжується, як описано в попередньому режимі роботи. Також у цьому випадку ймовірне походження коду відображається розробнику через вбудоване HTML-вигляд в IDE.

Попередня оцінка

Для перевірки ефективності запропонованого підходу передбачено оцінку точності у сценарії, де, маючи фрагмент коду, система повинна повертати релевантні посилання на потенційні джерела. Оцінювання повноти при цьому не здійснюється, оскільки повний перелік справжніх джерел для автоматично згенерованого коду зазвичай є невідомим, а також відсутні еталонні набори даних з позначеними джерелами, що дозволяли б виконати повну перевірку.

Для оцінки передбачається використати 112 завдань кодування (46 JavaScript та 55 Python) із набору даних HumanEval [8], для яких відомо принаймні одне релевантне посилання для автоматично згенерованого коду. Додатково планується вибрати 108 завдань (по 54 для кожної мови програмування) із бенчмарку MBPP [16], що містить приклади з реальних проєктів і зазвичай використовується для оцінки ефективності LLM у генерації коду [7].

Передбачається, що для кожного завдання спочатку AICertify згенерує код, який потім буде використано для формування запиту на отримання посилань через чат. Оцінка точності може здійснюватися шляхом незалежного аналізу вручну, де два оцінювачі перевіряють кожне посилання та узгоджують розбіжності.

Очікується, що система відображатиме лише ті посилання, які перевищують визначені пороги для метрик клонування та текстової подібності. Для різних значень порогів можна оцінити точність і середню кількість посилань, що будуть запропоновані користувачеві. Також планується перевірити можливість визначення ліцензії для кожного посилання, що дозволить користувачу оцінити, як код може бути повторно використаний або розповсюджений. При цьому слід враховувати, що не всі посилання дозволяють однозначно визначити ліцензію, особливо якщо вони вказують на блоги або посібники.

Висновки

У даній статті розглянуто проблему визначення походження програмного коду, згенерованого LLM. Запропоновано метод, що поєднує можливості мовних моделей, веб-пошуку та аналізу подібності програмного коду для встановлення потенційних джерел його походження.

Результати дослідження показують, що використання комбінованого підходу, який включає текстовий аналіз та виявлення клонів коду, дозволяє ефективно фільтрувати нерелевантні джерела та визначати можливі вебресурси, що містять схожі фрагменти програмного забезпечення.

Практична значущість запропонованого підходу полягає у можливості його використання для підвищення надійності програмних продуктів, створених із використанням LLM, а також для зменшення ризиків порушення авторських прав і ліцензійних умов.

Подальші дослідження можуть бути спрямовані на вдосконалення алгоритмів пошуку програмного коду, розширення методів автоматичного визначення ліцензій та створення інструментальних засобів підтримки запропонованого підходу у середовищах розроблення програмного забезпечення.

Перелік використаних джерел

[1] Extracting Training Data from Large Language Models / N. Carlini et al. *Proceedings of the 30-th USENIX Security Symposium*, 11–13 August 2021. Pp. 2633–2650.

[2] Quantifying Memorization Across Neural Language Models / N. Carlini et al. arXiv preprint. arXiv:2202.07646. 2023. Pp. 1–34. DOI: https://doi.org/10.48550/arXiv.2202.07646.

[3] Evaluating Large Language Models Trained on Code / M. Chen et al. arXiv preprint. arXiv:2107.03374. 2021. Pp. 3–14. DOI: https://doi.org/10.48550/arXiv.2107.03374.

[4] GitHub Copilot AI pair programmer: Asset or Liability? / A. M. Dakhel et al. *Journal of Systems and Software*. 2023. Vol. 203. Article 111734. DOI: https://doi.org/10.1016/j.jss.2023.111734.

[5] GitHub Copilot Documentation. URL: https://docs.github.com/en/copilot (дата звернення: 11.08.2025).

[6] Google Open Source. License Classifier. URL: https://github.com/google/licenseclassifier (дата звернення: 11.08.2025).

[7] Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation / Liu J., Xia C. S., Wang Y., Zhang L. arXiv preprint. arXiv:2305.01210. 2023. DOI: https://doi.org/10.48550/arXiv.2305.01210.

[8] CodeSearchNet Challenge: Evaluating the State of Semantic Code Search / H. Husain et al. arXiv preprint. arXiv:1909.09436. 2019. DOI: https://doi.org/10.48550/arXiv.1909.09436.

[9] On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? / Bender E. M., Gebru T., McMillan-Major A., Shmitchell S. *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 3–10 March 2021. Pp. 610–623. DOI: https://doi.org/10.1145/3442188.3445922.

[10] SPDX License List. Software Package Data Exchange (SPDX). URL: https://spdx.org/licenses/ (дата звернення: 11.08.2025).

[11] Training language models to follow instructions with human feedback / L. Ouyang et al. arXiv preprint. arXiv:2203.02155. 2022. Pp. 1–46. DOI: https://doi.org/10.48550/arXiv.2203.02155.
[12] Unveiling Memorization in Code Models / Z. Yang et al. arXiv preprint. arXiv:2308.09932. 2023. Pp. 1–11. DOI: https://doi.org/10.48550/arXiv.2308.09932.
[13] Kamiya T., Kusumoto S., Inoue K. CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering*. 2002. Vol. 28, no. 7. Pp. 654–670. DOI: https://doi.org/10.1109/TSE.2002.1019480.
[14] Deep Learning Meets Software Engineering: A Survey on Pre-Trained Models of Source Code / Niu C., Li C., Luo B., Ng V. *Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, Vienna, Austria, 23-29 July 2022. Pp. 5546–5555. DOI: https://doi.org/10.24963/ijcai.2022/775.
[15] Code Clone Detection based on Event Embedding and Event Dependency / Huang C., Zhou H., Ye C., Li B. arXiv preprint. arXiv:2111.14183. 2021. DOI: https://doi.org/10.48550/arXiv.2111.14183.
[16] Program Synthesis with Large Language Models / J. Austin et al. arXiv preprint. arXiv:2108.07732. 2021. Pp. 1–12. DOI: https://doi.org/10.48550/arXiv.2108.07732.

METHOD FOR DETERMINING THE PROBABLE ORIGIN AND LICENSE CONDITIONS OF PROGRAM CODE GENERATED BY LARGE LANGUAGE MODELS

Romanenko S.O. *senior lecturer, The Military Institute of Telecommunication and Information Technologies named after the Heroes of Kruty, Kyiv, ORCID: https://orcid.org/0009-0004-0240-0777, e-mail: serhi.romanenko@viti.edu.ua*

The active integration of large language models into the software development process is fundamentally changing approaches to code creation. One of the most common use cases for such models is the automatic generation of code fragments, which allows developers to increase productivity and reduce project implementation time. At the same time, the use of automatically generated code raises a number of new challenges, related to the lack of information about its origin, complicating the verification of source reliability, and potentially leading to copyright and licensing violations. This study addresses the problem of determining the origin of program code created with large language models and proposes an approach to solve it. The aim of the research is to develop a method for identifying probable sources of program code and determining possible usage licenses based on the analysis of similarity between generated code and fragments found in public sources. The proposed approach combines the capabilities of language models with web search mechanisms, followed by text similarity analysis and code clone detection. As a result, a list of web resources potentially containing similar code fragments is generated, and an attempt is made to automatically identify their licensing terms. The obtained results demonstrate the possibility of effectively filtering out irrelevant sources and increasing the transparency of using automatically generated software. The scientific novelty of this research lies in combining code similarity analysis methods with web search tools to determine its probable origin. The practical significance of the study is in the potential use of the proposed approach to enhance the reliability of software products and reduce the risks of license violations. A promising direction for future research is the improvement of code search algorithms and the expansion of methods for automatically determining licensing characteristics of code.
Keywords: *large language models; code generation; code provenance; code similarity analysis; software licensing; software engineering; artificial intelligence.*

References

[1] N. Carlini et al., “Extracting Training Data from Large Language Models,” in *Proc. of the 30-th USENIX Security Symposium*, August 11–13, 2021, pp. 2633–2650.
[2] Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramèr, and C. Zhang, “Quantifying Memorization Across Neural Language Models,” 2023, *arXiv:2202.07646*, pp. 1–34. **doi: 10.48550/arXiv.2202.07646**.
[3] M. Chen et al., “Evaluating Large Language Models Trained on Code,” 2021, *arXiv:2107.03374*, pp. 3–14. **doi: 10.48550/arXiv.2107.03374**.
[4] A. M. Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, “GitHub Copilot AI pair programmer: Asset or Liability?,” *Journal of Systems and Software*, vol. 203, article 111734, 2023. **doi: 10.1016/j.jss.2023.111734**.
[5] GitHub Copilot Documentation. [Online]. Available: https://docs.github.com/en/copilot. Accessed on: August 11, 2025.
[6] Google Open Source. License Classifier. [Online]. Available: https://github.com/google/licenseclassifier. Accessed on: August 11, 2025.
[7] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, “Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code

Generation," *arXiv:2305.01210*, 2023. **doi: 10.48550/arXiv.2305.01210**.

[8] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt, "CodeSearchNet Challenge: Evaluating the State of Semantic Code Search," 2019, *arXiv:1909.09436*. **doi: 10.48550/arXiv.1909.09436**.

[9] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?," in *Proc. of the 2021 ACM Conf. on Fairness, Accountability, and Transparency*, March 3–10, 2021, pp. 610–623. **doi: 10.1145/3442188.3445922**.

[10] PDX License List. Software Package Data Exchange (SPDX). [Online]. Available: https://spdx.org/licenses/. Accessed on: August 11, 2025.

[11] L. Ouyang et al., "Training language models to follow instructions with human feedback," 2022, *arXiv:2203.02155*, pp. 1–46. **doi: 10.48550/arXiv.2203.02155**.

[12] Z. Yang et al., "Unveiling Memorization in Code Models", 2023, *arXiv:2308.09932*, pp. 1–11. **doi: 10.48550/arXiv.2308.09932**.

[13] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: a multilinguistic token-based code clone detection system for large scale source code," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 654–670, 2002. **doi: 10.1109/TSE.2002.1019480**.

[14] C. Niu, C. Li, B. Luo, and V. Ng, "Deep Learning Meets Software Engineering: A Survey on Pre-Trained Models of Source Code," in *Proc. of the Thirty-First Int. Joint Conf. on Artificial Intelligence (IJCAI-22)*, Vienna, Austria, July 23-29, 2022, pp. 5546–5555. **doi: 10.24963/ijcai.2022/775**.

[15] C. Huang, H. Zhou, C. Ye, and B. Li, "Code Clone Detection based on Event Embedding and Event Dependency," 2021, *arXiv:2111.14183*. **doi: 10.48550/arXiv.2111.14183**.

[16] J. Austin et al., "Program Synthesis with Large Language Models," 2021, *arXiv:2108.07732*, pp. 1–12. **doi: 10.48550/arXiv.2108.07732**.

Стаття надійшла 29.01.2026
Стаття прийнята 01.03.2026
Стаття опублікована 26.03.2026

Цитуйте цю статтю як: Романенко С. О. Метод визначення ймовірного походження та ліцензійних умов програмного коду, згенерованого великими мовними моделями. *Вісник Приазовського державного технічного університету. Серія: Технічні науки*. 2026. Вип. 53, том 1. С. 54–60. DOI: https://doi.org/10.31498/2225-6733.53.1.2026.359775.