УДК 004.89
DOI https://doi.org/10.32782/2663-5941/2025.2.2/23

Павлов В. Г.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

СЕМАНТИЧНІ ТА ЛЕКСИЧНІ ОЗНАКИ РОЗРОБКИ ПРОГРАМНОГО КОДУ ШТУЧНИМ ІНТЕЛЕКТОМ

У статті розглянути проблеми, які виникають останній час у зв'язку зі масовим застосуванням штучного інтелекту (ШІ) для створення текстів, у тому числі й текстів комп'ютерних програм. Зазначено, що при загальному позитивному впливі, який насамперед зробив рутинну працю більш ефективною та швидкою, застосування штучного інтелекту створило й певні проблеми у сферах збереження авторського права та сприяння академічної доброчесності. Якщо раніше кожний випадок застосування творчих здобутків іншою людини розглядався як плагіат, який свідомо здійснив недоброчесний автор, то застосування при написанні штучного інтелекту зробило цей процес неявним та прихованим. Справа у тому, що будь-який штучний інтелект побудований на основі великої мовної моделі (LLM), яка розглядає елементи людської мови (токени), як числа, та обробляє їх за певними математичними формулами. Для цього оцифрування спочатку створюється певна множина токенів, які об'єднуються у послідовності. Таким чином відбувається генерація тексту. Але штучний інтелект передбачає постійне навчання за рахунок вивчення людських текстів. З них він бере й стали вирази, й цілі фрази, які потім застосовує при генерації. По факту відбувається створення текстів, які компілюються з багатьох джерел, але за рахунок використання чиїсь думок та здобутків.

У статті розглядаються сервіси, які виявляють застосування ШІ у звичайних текстах, робиться аналіз та порівняння їх ефективності на прикладі дослідження тестового тексту, який був згенерований ChatGPT. На підставі отриманих результатів автор визначає, що не завжди детектування дає достовірну інформацію про відсутність або наявність ознак застосування при створенні тексту генеративним ШІ.

Далі у статті описані дослідження з застосування ШІ для генерації текстів комп'ютерних програм. За результатами аналізу сервісів ШІ зроблена їх класифікація за мовами програмування та спеціалізацією сервісів. Подальші дослідження відбуваються зі застосуванням мови Assembler, яка найменше схожа з людською мовою. У статті наводяться приклади формулювання завдань для ШІ, за якими генеруються тексти програм. Далі аналізується якість та працездатність цих комп'ютерних програм, відмічаються певні ознаки, які дозволяють визначити, що текст програми був створений за допомогою ШІ. Також наведені результати детектування застосування ШІ при розробці комп'ютерних програм.

Ключові слова: *штучний інтелект, академічна доброчесність, детектування використання ШІ, Assembler, стиль тексту.*

Постановка проблеми. Сімдесят п'ять років тому відбулася подія, вплив якої зараз важко переоцінити: вийшла стаття відомого математика Алана Тюрінга [1, с. 454], у якої він розглядав можливість моделювання комп'ютером інтелектуальної діяльності людини. Там же у статті був вперше викладений славнозвісний «тест Тюрінга», у якому людині пропонувалося шляхом спілкування з іншою людиною та «розумною машиною», з'ясувати, де є спілкування з людиною, а де з її імітацією. На той час стаття була сприйнята з відомою долею гумору, бо ніхто, навіть у самих сміливих думках не міг передбачити, що імітація комп'ютерною програмою розумових дій людини колись стане реальністю. До речі, у тієї ж самої статті Тюрінг передбачив, що вдосконалення та розвиток імітаційних моделей буде відбуватися шляхом їх навчання.

Зараз ми можемо бачити, що застосування штучного інтелекту (ШІ) охопило майже усі сфери діяльності людини: медицину, транспорт, робототехніку, безпеку життєдіяльності, військову справу, офісну діяльність, бізнесову та фінансову діяльність та навіть деякі види мистецтва.

Але разом з тим виникає питання, чи спроможні ми контролювати розвиток штучного інтелекту, спрямовувати його застосування у потрібному напрямку та обмежувати там, де його безконтрольне використання може нанести шкоду у етичному сенсі. Насамперед мова йде про застосу-

вання штучного інтелекту у тих сферах, де творча діяльність людини регулюється такими категоріями, як інтелектуальна власність, авторське право, академічна доброчесність. Так, дійсно, генеративний штучний інтелект не є суб'єктом, якого можна розглядати, як дійсного автора якогось тексту, але він може вдало компілювати чиїсь твори, а тому його використання може розглядатися, як порушення авторського права тих осіб, чиї тексти він вдало переробив та застосував. Що стосується особи, яка буде користуватися цими «здобутками» генеративного ШІ, то залишимо кваліфікацію цієї дії на розсуд правників, нас буде цікавить інше питання, чи може бути виявлене застосування ШІ шляхом аналізу певних текстів? По суті це питання можна сформулювати як сучасну інтерпретацію «тесту Тюрінгу»: чи існують методи та засоби виявлення, хто дійсно є автором тексту: людина чи комп'ютерна програма. Цьому й будуть присвячені проведені дослідження.

Постановка завдання. Метою статті є викладення результатів проведених досліджень з визначення ознак застосування генеративного ШІ у звичайних текстах, а також у текстах комп'ютерних програм для перешкоджання неправомірного їх використання у якості свої здобутків не доброчесними користувачами.

Для досягнення означеної мети вирішуються наступні завдання:

– отримання примірників таких текстів для визначення можливостей генеративних ШІ;

– використання засобів детектування використання ШІ та порівняння їх ефективності;

– визначення семантичних та лексичних ознак застосування ШІ при аналізі текстів комп'ютерних програм.

Виклад основного матеріалу. Спочатку отримуємо об'єкт для дослідження. Найпростіший спосіб для цього – сформувати цей текст власноруч за допомогою якогось з сервісів штучного інтелекту. Їх достатньо багато, тому спочатку сформулюємо вимоги для, якім сервіс повинен задовольняти:

1. Спеціалізація з генерації текстів. Для цього сервіс повинен бути побудований на основі великих мовних моделей, тобто генерувати відповіді, шляхом побудови послідовності слів згідно лексичним особливостям певної людської мови. Зараз цьому критерію задовольняють більшість сервісів ШІ.

2. Підтримка української мови у запитах та відповідях. На жаль, ця вимога вельми суттєво скорочує перелік сервісів ШІ, бо, наприклад такі сервіси як **ClosersCopy** (https://www.closerscopy.com/), **AI-Writer** (https://ai-writer.com/) або **Writesonic** (https://writesonic.com/) не «розуміють» запитів українською та, відповідно, не генерують текстів українською мовою.

3. Безкоштовність або умовна безкоштовність. Ця вимога теж обмежила список сервісів, бо їх розробники у більшості розглядають їх як комерційні.

Після цього у залишку серед найвідоміших сервісів ШІ залишаються чат-боти **Gemini** від Google (***https://gemini.google.com/app***), **Claude** (***https://claude.ai***),**Copymate**(***https://copymate.app/***), **Copilot** (***https://www.copy.ai/***) та **ChatGPT** (***https://chatgpt.com***). Цікаво, що й **Copymate**, й **Copilot** базуються саме на **ChatGPT-4**, тому я майже без вагань обрав у якості полігона для досліджень саме **ChatGPT**. Після реєстрації та декількох тестових питань, на які я отримав досить розумну відповідь, я зробив запит до GhatGPT щодо можливості виявлення застосування ШІ при створенні текстового документу, на що отримав відповідь у вигляді тексту довжиной 295 слів, яка складалась зі 7 нумерованих пунктів та висновку.

За браком місця та тому, щ це – не мій текст, я приведу лише невеликий його фрагмент, у якому ChatGPT визначає ознаки застосування ШІ:

• *Проблеми із розумінням контексту. ШІ може створювати текст із правильними словами, але з помилковим або нерелевантним контекстом.*

• *Недостатня глибина аналізу. Алгоритми часто надають загальну інформацію без глибокого занурення у тему.*

Як бачимо, він надає цілком слушні поради та визначає деякі ознаки застосування самого себе, або схожих програм при створенні тексту. Як кажуть, йому, тобто **ChatGPT**, це краще знати зсередини цього процесу.

На другому етапі цей текст, який ПОВНІСТЮ був згенерований ШІ, досліджувався на декількох Інтернет-сервісах, які позначені, як детектори використання ШІ.

Усього було досліджено 5 сервісів, за допомогою яких виявляється застосування генеративного ШІ при формуванні документів. ТРИ з цих сервісів були запропоновані **ChatGPT** у відповіді на мій запит (**GPTZero, Originality.AI, Turnitin**. Ще ДВА сервіса (**zerogp, HIX.AI**) були знайдені мною самостійно під час виконання досліджень.

ДВА сервіси (**GPTZero, zerogp**) виявилися платними, але з можливістю використовувати їх безкоштовну версію, звичайно, з певними обмеженнями. У одному сервісі (**Originality.AI** –

https://originality.ai/) можливість працювати безкоштовно не була мною знайдена. Один сервіс (**Turnitin** – ***https://www.turnitin.com/***) потребував ліцензії, а один (**HIX.AI**) працює ЦІЛКОМ БЕЗКОШТОВНО.

Тому фактично вдалося протестувати лише ТРИ сервіси. При цьому:

• сервіс **zerogp** (***https://www.zerogpt.com/***) надав дуже НЕТОЧНУ інформацію (**32,4 %**) при тестуванні документа, який був ПОВНІСТЮ створений за допомогою ChatGPT;

• сервіс **GPTZero** (***https://gptzero.me/***) надав значну кращу оцінку (**97 %**), причому з поясненнями щодо цього, які до того ж надані у вигляді звіту у форматі PDF-файлу (доданий до модулю);

• сервіс **HIX.AI** (***https://bypass.hix.ai/ru/originality-ai***) не зробив висновку у вигляді цифр, але текстово повідомив, що наданий текст скоріше за все був згенерований за допомогою ШІ. Також цей сервіс надав можливість перевірити ще один документ. В якості нього був наданий фрагмент тексту з цього модулю, який я написав самостійно. Цей фрагмент був перевірений з результатом, текст на 100 % написаний людиною, що відповідає дійсності.

Таким чином, перший етап досліджень показав, що не усі сервіси, які декларують себе, як таки, що виявляють наявність у тексті ознак використання ШІ, насправді можуть це робити з достатньою точністю. Більш-менш впорались з цим завданням **GPTZero** та **HIX.AI**.

Наступним етапом досліджень була перевірка спроможності ШІ генерувати тексти комп'ютерних програм та також виявляти цей факт. Згадаємо, що ще Алан Тюринг передбачав, що це стане колись можливим [1, с. 455].

Дійсно, зараз існую досить велика кількість ШІ-сервісів, які декларують про свої спроможності у цьому напрямку. І це не дивно, якщо згадати, що будь-яка мова програмування побудована за принципами формальної граматики, коли при побудові ланцюжки слів (команд у програмі) рівень передбачення наступного слова (команди) є набагато більшим, чим у природній людський мові. Дійсно, у кожній мові програмування існують стали синтаксичні ланцюжки як всередині складних команд, так й між ними. Інша справа у тому, що на відміну від висловлювань на людський мові, де не завжди присутня конкретика, мова програмування призначена для вирішення у межах програми певної задачі. Тобто, якщо програма цю задачу не вирішує та не надає вірну відповідь, то вона ні до чого.

Знову ж таки для визначення кола безкоштовних сервісів, які генерують тексти комп'ютерних

Таблиця 1

Сервіси ШІ, які генерують тексти комп'ютерних програм

	Python	**JavaScript**	**C++**	**Java**	**Assembler**
Онлайн-асистенти штучного інтелекту	ChatGPT Google Gemini Phind Hugging Face CodeGen	ChatGPT Google Gemini Phind Hugging Face CodeGen	ChatGPT Google Gemini Phind Codeium	ChatGPT Google Gemini Phind Codeium	ChatGPT Google Gemini Phind DeepAI
Інструменти автогенерації коду	Codeium Replit Ghostwriter AWS CodeWhisperer	Codeium Replit Ghostwriter Tabnine	Replit Ghostwriter Tabnine AWS CodeWhisperer	Replit Ghostwriter Tabnine AWS CodeWhisperer	Replit Ghostwriter Tabnine AWS CodeWhisperer
No-Code та Low-Code платформи	Anvil Google Colab Streamlit	Bubble Glide OutSystems	Snap! Blockly	OutSystems AppSheet	–
Відкриті AI-інструменти та моделі	GPT4All StarCoder Codex	GPT4All StarCoder Codex	GPT4All StarCoder Hugging Face CodeGen	GPT4All StarCoder Hugging Face CodeGen	GPT4All StarCoder Hugging Face CodeGen
Автоматизація робочих процесів	Zapier AI Make	Zapier AI Make	Zapier AI Make	Zapier AI Make	Zapier AI Make
Онлайн-редактори з AI-допомогою	–	JSFiddle CodePen StackBlitz	–	–	–
Онлайн-компілятори з AI-допомогою	–	–	OnlineGDB JDoodle CodeChef IDE	OnlineGDB JDoodle OneCompiler	ASM80 Online Assembler IDE ASy68K (емулятор для Motorola 68000

програм вдамося до допомоги ChatGPT. Спочатку був зроблений запит відносно переліку таких сервісів, але їх виявилось чимало, тому потім запит було кореговано у бік позначення мов програмування, на яких «спеціалізуються» сервіси ШІ. За підсумками була складена таблиця зі розбивками за мовами програмування та призначенням сервісів (Табл. 1).

З переліку мов, які були у попередньому списку виберемо **Python, JavaScript, C++** та **Java**. Оскільки мене цікавить спроможність штучного інтелекту створювати програми на мовах низького рівня, то додаємо у цей перелік також **Assembler**.

При цьому ChatGPT сформулював за моїм запитом, як саме треба надавати завдання сервісу ШІ щодо написання тексту програми:

Щоб отримати найкращий результат, слід чітко вказати:

- ***Призначення програми:*** *Наприклад, «створити калькулятор витрат».*
- ***Основні функції:*** *«Користувач вводить дані, зберігає їх у базі, виводить графік».*
- ***Бажана мова програмування:*** *«Python + Flask», «JavaScript + React», тощо.*
- ***Цільова платформа:*** *«Мобільний додаток», «Веб-сайт», «Десктоп-додаток».*

Як бачимо, нічого складного, щоб отримати готовий текст програми, немає, але спробуємо поставити більш конкретну задачу, у якої буде присутній розрахунок за певною формулою. Спочатку задамо таке формулювання умов задачі:

> Напиши програму на мові Assembler x86 (32-біт, NASM), яка обчислює площу прямокутника за формулою: $S = a * b$
>
> Значення a та b вводяться користувачем, результат виводиться у консоль

На це ШІ відповів згенерованою програмою довжиною у 39 рядків та ще декількома рядками коментарів. Але програма була не дуже зручною для користування, хоча й мала вигляд робочої. Тоді завдання було пере-формульовано таким чином:

> Напиши програму на мові Assembler x86 (MASM32), яка обчислює площу прямокутника за формулою: $S = a * b$ Значення a та b задаються користувачем, результат виводиться у вікно за допомогою функції MessageBox.

На цей раз у відповіді ШІ був програмний код довжиною у 31 рядка та ще майже стільки же рядків з коментарями у тексті програми. Також окремо після тексту програми були надані рекомендації щодо компіляції цієї програми та її виконання у середовищі Assembler.

Загальна структура програми була також вірною:

- директиви компілятору щодо створення середовища;
- підключення зовнішніх файлів з бібліотеками та макросами;
- початкова ініціалізація змінних у секції ***.data***;
- рядки з командами у секції ***.code***;
- початкова мітка та завершуючи команда ***end <мітка>***;
- вірне застосування функції ***MessageBox***() та ***ExitProcess***() з усіма необхідними параметрами.

Але були певні ознаки, які можна віднести до тих, які можуть позначати застосування ШІ при генерації цього коду. До таких можна віднести:

- наявність надмірно великої кількості коментарів, що впадає у очі;
- підключення зовнішніх файлів без вказування відносного шляху до них, що вірно, якщо програмний файл створюється безпосередньо у теки з файлами Assembler, але вважається не дуже добрим, бо використовувати для зберігання своїх файлів теку програми не вважається коректним;
- застосування для попереднього перетворення числових значень у текст та навпаки достатньо «екзотичних» функції ***atodw*()** та ***dwtoa*()** замість більш вживаної функції ***wsprintf*()**, яка використовує один буфер замість декількох. До того ж опис функції ***StdIn*()**, ***atodw*()** та ***dwtoa*()** не міститься у вказаних спочатку файлів імпортованих макросів та бібліотек **windows.inc**, **user32.inc**, **kernel32.inc**, **user32.lib** та **kernel32.lib**, тому при компіляції виникла помилка, пов'язана з неможливістю компілятора розпізнати вказані функції. Тому треба було ДОДАТКОВО підключати необхідні файли **masm32.inc** та **masm32.lib**, після чого програма коректно СКОМПІЛЮВАЛАСЯ та ВИКОНАЛАСЯ;
- для введення значень використовувалася функція ***StdIn*()**.

Можна констатувати, що в цілому програма була складена достатньо грамотно та могла бути використана при потребі, як свій «здобуток», звичайно з умовою виправлення усіх наведених помилок.

Але ця програма призначена для консольного вводу, тому при зчитуванні з консолі треба ще отримати дескриптор стандартного вводу. Щоб уникнути цього спробуємо вказати у запиті

Таблиця 2

Вірогідність застосування ШІ у розробці текстів комп'ютерних програм

	Програма-1	Програма-2	Програма-3	Програма-4	Програма-5
zerogpt (https://www.zerogpt.com/)	0 %	0 %	0 %	0 %	37,89 %
GPTZero (https://gptzero.me/)	87 %	4 %	4 %	91 %	4 %

до **ChatGPT**, що значення змінних **a** та **b** повинні задаватися безпосередньо у програмі, як константи:

> Напиши програму на мові Assembler x86 (MASM32), яка обчислює площу прямокутника за формулою: S = a * b Значення a та b задаються у програм, як константи, результат виводиться у вікно за допомогою функції MessageBox

На цей раз у якості відповіді отримаємо нову версію програми, де, дійсно, значення змінних ***a*** та ***b*** вже не потрібно вводити з клавіатури, оскільки вони задаються безпосередньо у програмі.

Ця версія програми містить ті ж недоліки з не зовсім коректним підключенням бібліотек, імпортованих зі зовнішніх файлів та відсутністю певних бібліотек, що й приклад вище, але вже відсутня необхідність виклику консольного вікна, що добре.

Як бачимо, навіть без застосування програмних детекторів, програми, які згенеровані ШІ, мають деякі ознаки стилю, як у наборі використаних функції, так й в оформленні самої програми, тобто ознаки, які відносяться до семантики та синтаксису тексту програми, який згенерував ШІ.

На завершенні досліджень для детектування використання ШІ у текстах комп'ютерних програм були застосовані ті ж сервіси, що були успішно застосовані для аналізу фрагменту звичайного тексту у попередньому дослідженні, а саме: **zerogpt** (https://www.zerogpt.com/), **GPTZero** (https://gptzero.me/) та **HIX.AI** (https://bypass.hix.ai/ru/originality-ai).

Для тестування був сформований пакет з 5 текстів комп'ютерних програм, серед яких були як програми, які були написані людиною (програми за номерами 2, 3 та 5), так й тексти, які були згенеровані за допомогою штучного інтелекту у вигляді чат-боту ChatGPT (https://chatgpt.com).

На жаль, для одного зі сервісів, а саме **HIX.AI** попередньо була доступна перевірка невеликих (до 125 слів) фрагментів, але, мабуть ліміт вже виявився вичерпаним, тому доступна тільки його використання на **платній** основі.

Результати перевірки за допомогою решти сервісів зведені у таблицю.

За підсумками перевірки ефективності детектування текстів коп'ютерних програм можна констатувати, що сервіс **GPTZero** впорався з детектуванням на «відмінно», бо виявив високу вірогідність застосування ШІ при розробці текстів комп'ютерних програм саме там, де він був присутній, а решта контрольних прикладів була детектована, як така, що найбільш ймовірно створена людиною, що також **абсолютно вірно**. Сервіс **zerogpt**, на жаль продемонстрував недостатню ефективність у детектуванні, тому його подальше використання у мене викликало багато питань.

Висновки. За підсумками проведених досліджень з використання ШІ для генерації звичайних текстів та текстів комп'ютерних програм з їх наступним детектуванням можна констатувати наступне:

Штучний інтелект, а саме **ChatGPT**, цілком спроможні генерувати правдоподібні звичайні тексти та тексти комп'ютерних програм на різних мовах програмування, зокрема на мові **Assembler**, яка найменш схожа з людською мовою.

На підставі аналізу згенерованих текстів програм на мові Assembler виявлені певні семантичні та синтаксичні ознаки використання для їх створення штучного інтелекту.

Використання низки сервісів для детектування застосування ШІ для генерації як звичайних текстів, так й текстів комп'ютерних програм показало, що не усі вони надають достовірні результати, але деяким можна довіряти більше.

Список літератури:

1. Turing A. M. Computing Machinery and Intelligence. *Mind. Oxford University Press*, 1950. Vol. 59, No. 236. P. 433–460. DOI: https://doi.org/10.1093/mind/LIX.236.433

Pavlov V. G. SEMANTIC AND LEXICAL FEATURES OF PROGRAM CODE DEVELOPMENT BY ARTIFICAL INTELLIGENCE

The article discusses the problems that have arisen recently in connection with the massive use of artificial intelligence (AI) for creating texts, including texts of computer programs. It is noted that despite the overall positive impact, which primarily made routine work more efficient and faster, the use of artificial intelligence also created certain problems in the areas of copyright preservation and promotion of academic integrity. If earlier each case of the use of creative achievements by another person was considered as plagiarism, which was deliberately committed by a dishonest author, then the use of artificial intelligence in writing made this process implicit and hidden. It is the fact that any artificial intelligence is built on the basis of a large language model (LLM) that considers the elements of human speech (tokens) as numbers and processes them according to certain mathematical formulas. For this digitization, at first, a certain set of tokens is created, which are combined in sequences. In this way, text generation occurs. But artificial intelligence involves continuous learning through the study of human texts. From them, it takes both permanent expressions and whole phrases, which it then uses when generating. In fact, texts are created that are compiled from many sources, but at the expense of the use of someone's opinions and achievements.

The article discusses services that detect the use of AI in plain texts, analyzes and compares their effectiveness using the example of a test text study, that was generated by ChatGPT. Based on the results obtained, the author determines that detection does not always provide reliable information about the absence or presence of signs of use in text creation by generative AI.

The article further describes research on the use of AI to generate computer program texts. Based on the results of the analysis of AI services, they are classified by programming languages and specialization of services. Further research takes place using the Assembler language, which is the least similar to human language. The article provides examples of the formulation of tasks for AI, according to which program texts are generated. Further, the quality and performance of these computer programs are analyzed, certain signs are noted that make it possible to determine that the text of the program was created using AI. The results of detecting the use of AI in the development of computer programs are also presented.

Key words: *artificial intelligence, academic integrity, detection of AI using, Assembler, writing style.*