https://doi.org/10.31891/2219-9365-2026-87-52
УДК 004.42:004.93'12

НЕМОВ Руслан
Черкаський державний технологічний університет
https://orcid.org/0009-0006-3219-450X
r.h.nemov.asp23@chdtu.edu.ua

КІЛЬКІСНА ОЦІНКА ІНФОРМАТИВНОСТІ СЛОВНИКА ОЗНАК ДЛЯ КЛАСИФІКАЦІЇ ПРОГРАМНИХ КОДІВ ЛЮДЕЙ ТА ГЕНЕРАТИВНИХ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

Розглянуто задачу кількісного оцінювання інформативності словника ознак для класифікації програмних кодів за умов поширення генеративних моделей штучного інтелекту (ШІ). Запропоновано нову метрику інформативності словника – кількість моделей-класифікаторів у турнірі, що досягають 100% точності класифікації на рівні вікон та на рівні класів. Метрика розглядається як доповнення до традиційних показників (точність найкращої моделі, середня точність турніру) і характеризує не одну окрему модель, а розподіл якості за усім набором архітектур турніру. Експериментально на датасеті з 12 авторів (10 людей та 2 моделі ШІ) у 4 мовах програмування досліджено 22 серії у трьох сценаріях: внутрішньомовний (4 серії), крос-мовний (10 серій) та розрізнення Людина/ШІ (8 серій). Показано, що додавання символьних крос-зв'язків до базового словника ознак збільшує кількість моделей зі 100% точністю на рівні класів у середньому на +3,23 моделі на серію. Найбільший приріст виявлено у внутрішньомовному сценарії (+6,25 моделі) та у задачі розрізнення Людина/ШІ (+4,38 моделі). Запропонована метрика забезпечує більш детальну характеристику інформативності словника ознак, аніж агрегована середня точність.

Ключові слова: інформативність словника ознак, атрибуція авторства коду, виявлення коду генеративного ШІ, турнір моделей-класифікаторів, інтелектуальний моніторинг.

NEMOV Ruslan
Cherkasy State Technological University

QUANTITATIVE ASSESSMENT OF FEATURE DICTIONARY INFORMATIVENESS FOR CLASSIFICATION OF SOURCE CODE WRITTEN BY HUMANS AND GENERATIVE ARTIFICIAL INTELLIGENCE MODELS

The article addresses the task of quantitatively assessing the informativeness of a feature dictionary for source code classification under the widespread use of generative artificial intelligence (AI) models such as ChatGPT and Claude Code. Existing approaches evaluate dictionary quality through aggregated metrics – best-model accuracy or mean accuracy across the tournament of classifier models – which characterise either a single specific architecture or an averaged value but do not reflect the distribution of model quality across the entire tournament. To overcome this limitation, a novel informativeness metric is proposed: the number of classifier models in the tournament that achieve 100% classification accuracy at the window level and at the class level. This metric is positioned as a complement to traditional metrics and characterises not a single model but the distribution of quality across the entire set of tournament architectures. The proposed metric is grounded in the input-data-array (IDA) formation methodology of the S.V. Holub research school, originally developed for Ukrainian text classification and extended in this work to source code classification. Experimentally, on a dataset of 12 authors (10 humans and 2 AI models) across 4 programming languages (Java, JavaScript, TypeScript, Python), 22 series were investigated in three scenarios: within-language (4 series), cross-language (10 series), and Human/AI discrimination (8 series). A tournament of 39 classifier model architectures including Dense, CNN, LSTM, GRU, BiLSTM, Attention, BERT, and the Group Method of Data Handling (GMDH) was used. It is shown that the addition of symbolic cross-links between words to the baseline feature dictionary increases the number of models with 100% accuracy at the class level by +3.23 models per series on average. The largest gain is observed in the within-language scenario (+6.25 models) and in the Human/AI discrimination task (+4.38 models). The proposed metric provides a more detailed characterisation of feature dictionary informativeness than aggregated mean accuracy, which is particularly important for selecting effective feature sets in source code authorship attribution.

Keywords: feature dictionary informativeness, source code authorship attribution, generative AI code detection, classifier model tournament, intelligent monitoring.

Стаття надійшла до редакції / Received 10.05.2026
Прийнята до друку / Accepted 10.07.2026
Опубліковано / Published 10.09.2026

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

Стрімкий розвиток генеративних моделей штучного інтелекту (ШІ) – ChatGPT, Claude Code, Gemini, Copilot та інших – призвів до того, що значна частина програмного коду тепер генерується автоматично, а не пишеться вручну розробниками. Виникла практично актуальна задача автоматичного розрізнення коду людини від коду, згенерованого моделлю ШІ. Сферами застосування цієї задачі є виявлення плагіату в академічному середовищі, проведення судової програмно-технічної експертизи, аналіз потенційних загроз в

екосистемах відкритого коду та підтримка академічної доброчесності в умовах поширення інструментів автоматичної генерації коду.

Розв'язання цієї задачі потребує формування інформативного словника ознак, на основі якого моделі-класифікатори здатні розрізняти стиль різних авторів. Однак на сьогодні відсутні універсальні підходи до кількісного оцінювання якості сформованого словника ознак (масиву вхідних даних, МВД). Існуючі підходи зазвичай характеризують ефективність словника через одну з двох метрик: точність найкращої моделі-класифікатора або середню точність по всьому турніру моделей. Жоден із цих підходів не відображає розподілу якості моделей у турнірі: дві серії з однаковою середньою точністю можуть мати принципово різний розподіл (одна – багато слабких моделей з кількома сильними, інша – однорідний середній рівень). Тому актуальною є розробка кількісної метрики, яка характеризує інформативність словника через розподіл якості моделей.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Стилометрія програмних кодів як галузь активно розвивається з 1990-х років. Серед найбільш цитованих робіт – дослідження Caliskan-Islam та співавторів [1], яке продемонструвало можливість деанонімізації авторів коду через їх стиль на платформі Google Code Jam, досягши 95% точності для 1600 авторів. Метод базувався на витяганні комплексних ознак з абстрактних синтаксичних дерев (AST), лексичних характеристик і просторових патернів. Burrows та співавтори [2] провели систематичний огляд методів стилометрії програмних кодів. З розвитком глибокого навчання Abuhamad та співавтори [3] запропонували масштабовану систему ідентифікації авторства на основі рекурентних нейронних мереж. Abuhamad та співавтори [4] застосували поєднання рекурентних нейронних мереж та методу випадкового лісу з глибинним навчанням ознак для широкомасштабної атрибуції коду на C++, Java та Python.

Окрему течію складають embedding-моделі. CodeBERT [5] та GraphCodeBERT дозволяють отримувати векторні представлення фрагментів коду. Однак усі ці підходи характеризуються мовною залежністю – модель, навчена на одній мові програмування, не переноситься на інші без значного перенавчання. З появою великих мовних моделей актуальною стала також задача розрізнення людського і машинно-згенерованого коду. У роботах [6, 7] досліджуються методи виявлення коду, згенерованого ШІ, переважно з використанням специфічних для конкретних моделей характеристик.

Наукова школа С.В. Голуба [8, 9] розробила методологію формування масиву вхідних даних (МВД) для класифікації україномовних текстів у структурі інформаційної технології інтелектуального моніторингу. Ключовими елементами методології є: розбиття вхідного тексту на вікна фіксованої довжини, обчислення набору чисельних ознак для кожного вікна, застосування ймовірнісного критерію інформативності з межею інформативної достатності (МІД) для відсіювання не інформативних ознак та формування адаптивного словника ознак. У даній роботі цей підхід поширюється на програмні коди шляхом введення нового типу ознак – символьних крос-зв'язків між словами. Запропоновано також двопоказникове оцінювання ефективності методу через дві агреговані метрики: точність найкращої моделі турніру та середню точність по всіх моделях.

Невирішеною залишається проблема відсутності метрики, яка враховує розподіл якості моделей у турнірі: ні точність найкращої моделі, ні середня точність турніру не дають інформації про те, скільки конкретно моделей у турнірі досягають високого рівня класифікації. Така метрика особливо важлива для практичних задач, де потрібно обирати ансамбль моделей з близькою високою точністю.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою статті є розробка кількісної метрики оцінювання інформативності словника ознак для класифікації програмних кодів та її експериментальна апробація для атрибуції авторства коду людей і генеративних моделей ШІ. Завдання роботи такі: (1) запропонувати метрику оцінювання інформативності словника ознак на основі кількості моделей у турнірі, що досягають 100% точності класифікації; (2) формалізувати запропоновану метрику в межах методології школи С.В. Голуба; (3) виконати експериментальне дослідження на датасеті з 12 авторів (10 людей та 2 моделі ШІ) у 4 мовах програмування; (4) проаналізувати результати у трьох сценаріях – внутрішньомовному, крос-мовному та сценарії розрізнення Людина/ШІ.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Методологічна основа

У роботі застосовано методологію формування масиву вхідних даних (МВД) школи професора С.В. Голуба [8, 9], розроблену для класифікації україномовних текстів і поширену на програмні коди. Задача класифікації авторства програмних кодів формалізована таким чином. Дано скінченну множину програмних кодів

$$T = \{t_1, t_2, ..., t_n\} \tag{1}$$

експертно згруповану за авторством у m класів множини K:

$$K = \{k_1, k_2, ..., k_m\} \tag{2}$$

де m – кількість авторів. Необхідно побудувати модель-класифікатор f, що відображає елементи нової множини T* (нових кодів після навчання, $T^* \notin T$) на елементи множини K:

$$f : T^* \to K \tag{3}$$

Для фільтрації ознак при формуванні МВД застосовується ймовірнісний критерій інформативності з межею інформативної достатності (МІД, параметр T у відсотках):

$$P_i = \frac{V_i}{n} \cdot 100\% \tag{4}$$

де V_i – кількість використань і-ї ознаки у вікні, n – загальна кількість знаків у вікні. Ознака потрапляє до адаптивного словника МВД, якщо її показник P_i перевищує значення МІД (T %). Базовий словник містить ознаки наступних типів: базові метрики вікна (середня довжина слова, кількість рядків), частоти n-грам символів довжиною від 1 до 4, позиційні ознаки слів.

Запропонована метрика – кількість моделей із 100% точністю

У даній роботі застосовується двопоказникове оцінювання ефективності методу через дві агреговані метрики: точність найкращої моделі турніру та середню точність по всіх моделях. Однак ці метрики не дозволяють судити про розподіл якості моделей у турнірі. У даній статті пропонується доповнити двопоказникове оцінювання новою метрикою – кількістю моделей у турнірі, що досягають 100% точності класифікації. Метрика обчислюється окремо для рівня вікон і рівня класів:

$$M_W^{100} = \left|\{m \in M : acc_W(m) = 100\%\}\right| \tag{5}$$

$$M_c^{100} = \left|\{m \in M : acc_c(m) = 100\%\}\right| \tag{6}$$

де M_W^{100} – кількість моделей зі 100% точністю на рівні вікон, M_c^{100} – кількість моделей зі 100% точністю на рівні класів, $acc_W(m)$ – точність моделі m на рівні вікон, $acc_c(m)$ – точність моделі m на рівні класів після мажоритарного голосування вікон одного класу, M – загальний турнір моделей.

Перевага запропонованого методу формування словника обчислюється як різниця цих показників між варіантом з крос-зв'язками та базовим варіантом:

$$\Delta M^{100} = M_{cross}^{100} - M_{base}^{100} \tag{7}$$

Додатне значення ΔM свідчить про збільшення кількості «відмінних» моделей у турнірі після додавання крос-зв'язків, що інтерпретується як покращення інформативності словника ознак. Принципова перевага запропонованої метрики – вона характеризує не одну окрему модель і не агреговане середнє, а саме розподіл якості за всім набором архітектур турніру. Збільшення кількості моделей зі 100% точністю означає, що інформативність словника достатня для більшої кількості різних архітектур, що забезпечує надійність ансамблевого підходу.

Експериментальне дослідження

Для експериментів сформовано датасет, що містить програмні коди 12 авторів. З них 10 – люди-розробники з відкритих репозиторіїв GitHub: DmytroDanylyk (Java), JoshuaBloch (Java), VladKravets (Java), DmytroSemenov (JS), VolodymyrAgafonkin (JS), PaulMiller (TS), VladShatskiy (TS), TarasMaichuk (C#/TS), MaxKotliar (Python), SergheiIakovlev (Python). Решта 2 автори – генеративні моделі ШІ: ChatGPT (OpenAI) та Claude Code (Anthropic). Коди ШІ-моделей згенеровано від початку до кінця в одній сесії за технічним завданням, без подальшого редагування людиною.

Параметри МВД зафіксовано на наступних значеннях: розмір вікна N = 500 знаків, межа інформативної достатності T = 5%, режим фільтрації ознак – AVG. Турнір моделей містить 39 архітектур, серед яких представники наступних сімейств: повнозв'язні мережі Dense (з різними функціями активації), згорткові CNN_1D, рекурентні LSTM, GRU та BiLSTM з механізмами уваги, трансформер-подібні з Attention-шарами, автоенкодери, embedding-моделі (NNLM, Universal Sentence Encoder), BERT для класифікації послідовностей та метод групового врахування аргументів (МГУА) Івахненка [12]. Усі моделі навчалися з ідентичними гіперпараметрами на одному й тому самому МВД.

Дослідження проведено в трьох сценаріях. У внутрішньомовному сценарії (4 серії, блок 1) «своїм» автором послідовно виступає один із Java-авторів, «чужими» – інший Java-автор. У крос-мовному сценарії (10 серій, блок 5) «своїм» виступає кожен з 10 людських авторів, «чужими» – інші 9. У сценарії розрізнення Людина/ШІ (8 серій, блок 6) «своїми» послідовно виступають моделі ШІ та люди-Java-розробники, «чужими» – представники протилежного класу. Для кожної серії проведено два запуски: базовий (МВД без крос-зв'язків) і з крос-зв'язками.

Результати дослідження

Результати по 22 серіях наведено в табл. 1, 2, 3 (по сценаріях). У кожній таблиці для кожної серії вказано кількість моделей із 39, що досягли 100% точності за вікнами (M100w) та за класами (M100c) у базовому варіанті та у варіанті з крос-зв'язками, а також різниця ΔM_W і ΔM_c.

Таблиця 1

Результати у внутрішньомовному сценарії (Java vs Java)

Серія	«Свій»	Всього моделей	M100 (вікна)		M100 (класи)		Ознак
			Базовий	З крос	Базовий	З крос	
1.1	DmytroDanylyk	39	4	4	3	14	29 / 108
1.2	JoshuaBloch	39	1	2	1	1	31 / 159
1.3	VladKravets	39	4	0	3	14	30 / 114
1.5	ClaudeCode	39	0	0	7	10	27 / 77
Сер.	—	39	2,25	1,50	3,50	9,75	—

Примітка: M100 – кількість моделей із 39 у турнірі, які досягли 100% точності класифікації; «Ознак» – кількість ознак у словнику (Базовий / З крос-зв'язками).

У внутрішньомовному сценарії додавання крос-зв'язків стабільно збільшує кількість моделей зі 100% точністю на рівні класів (середній приріст +6,25 моделі на серію). На рівні вікон ефект менш виражений (середня зміна −0,75). Найбільший приріст спостерігається у серіях 1.1 (DmytroDanylyk: +11 моделей по класах) та 1.3 (VladKravets: +11 моделей по класах). Це пояснюється тим, що крос-зв'язки збільшують кількість ознак словника у 2,6–5,1 разів (з 27–31 до 77–159 ознак), що дозволяє більшій кількості різних архітектур моделей вловлювати стилістичні особливості автора.

Таблиця 2

Результати у крос-мовному сценарії (один автор проти 9 інших)

Серія	«Свій» (мова)	Всього моделей	M100 (вікна)		M100 (класи)		Ознак
			Базовий	З крос	Базовий	З крос	
5.1	DmytroDanylyk (Java)	39	0	0	11	22	29 / 108
5.2	DmytroSemenov (JS)	39	0	0	20	8	35 / 193
5.3	JoshuaBloch (Java)	39	0	0	0	0	31 / 159
5.4	MaxKotliar (Python)	39	0	0	5	9	— / —
5.5	PaulMiller (TS)	39	0	0	11	28	38 / 235
5.6	SergheiIakovlev (Python)	39	0	0	2	0	— / —
5.7	TarasMaichuk (C#/TS)	39	0	0	0	0	— / —
5.8	VladKravets (Java)	39	0	0	7	7	30 / 114
5.9	VladShatskiy (TS)	39	0	0	1	0	— / —
5.10	VolodymyrAgafonkin (JS)	39	0	0	6	0	38 / 268
Сер.	—	39	0,00	0,00	6,30	7,40	—

У крос-мовному сценарії жодна модель турніру не досягає 100% точності на рівні вікон (оскільки вибірка містить 825-853 вікон з 100-104 класами, тобто завдання значно складніше). На рівні класів спостерігається помірне зростання середньої кількості «відмінних» моделей (+1,10 моделі на серію). Найвищі прирости – у серіях 5.5 (PaulMiller TS, +17 моделей) та 5.1 (DmytroDanylyk Java, +11 моделей). Негативні результати мають серії 5.2 (DmytroSemenov, −12), 5.10 (VolodymyrAgafonkin, −6) – у цих випадках додавання крос-зв'язків зменшує кількість «відмінних» моделей, що може свідчити про надмірну специфічність нових ознак до тренувальної вибірки.

Таблиця 3

Результати у сценарії розрізнення Людина/ШІ

Серія	«Свій» / «Чужі»	Всього моделей	M100 (вікна)		M100 (класи)		Ознак
			Базовий	З крос	Базовий	З крос	
6.1	ChatGPT / DmytroDanylyk	39	1	1	1	13	32 / 147
6.2	ChatGPT / JoshuaBloch	39	3	1	3	13	32 / 147
6.3	ClaudeCode / DmytroDanylyk	39	0	0	5	12	27 / 77
6.4	ClaudeCode / JoshuaBloch	39	0	0	0	3	27 / 77
6.5	DmytroDanylyk / ChatGPT	39	2	4	7	17	29 / 108
6.6	DmytroDanylyk / ClaudeCode	39	13	0	13	19	29 / 108
6.7	JoshuaBloch / ChatGPT	39	17	6	8	1	31 / 159
6.8	JoshuaBloch / ClaudeCode	39	7	1	7	1	31 / 159
Сер.	—	39	5,38	1,63	5,50	9,88	—

У сценарії розрізнення Людина/ШІ середній приріст кількості «відмінних» моделей за класами становить +4,38 моделі на серію. Особливо яскраві позитивні результати – у серіях, де ШІ виступає «своїм» (6.1, 6.2, 6.3): +12...+13 моделей. Це свідчить, що крос-зв'язки добре характеризують специфіку ШІ-згенерованого коду. Серії 6.7 та 6.8 (JoshuaBloch як «свій») показують негативний результат, що пояснюється високою стилістичною уніфікованістю стилю цього автора (автор книги «Effective Java»), через що додавання крос-зв'язків вносить шум.

Таблиця 4

Підсумкова зміна кількості моделей із 100% точністю

Сценарій	Серій	Δ M100 (вікна)	Δ M100 (класи)
Внутрішньомовний (Java vs Java)	4	−0,75	+6,25
Крос-мовний	10	0,00	+1,10
Розрізнення Людина/ШІ	8	−3,75	+4,38
Усього	22	−1,50	+3,23

Аналіз та інтерпретація результатів

Запропонована метрика виявляє кілька важливих закономірностей, не помітних при агрегованому оцінюванні. По-перше, додавання символьних крос-зв'язків у середньому збільшує кількість моделей із 100% точністю на рівні класів (+3,23 моделі на серію по 22 серіях). При цьому на рівні вікон спостерігається невелике зменшення (−1,50). Це означає, що крос-зв'язки покращують саме результати після мажоритарного голосування, тобто саме на тому рівні, який релевантний для практичної атрибуції.

По-друге, ефект крос-зв'язків залежить від сценарію. Найбільший приріст – у внутрішньомовному сценарії (+6,25 моделі на серію за класами), тобто коли йдеться про розрізнення авторів однієї мови. У сценарії розрізнення Людина/ШІ – +4,38, що теж значний результат. У крос-мовному сценарії приріст невеликий (+1,10), що пояснюється значно складнішою задачею (10 класів проти 100 класів) і зниженою інформативністю будь-яких ознак при такій кількості «чужих» класів.

По-третє, негативні результати у деяких серіях (5.2, 5.10, 6.7, 6.8) виявляють важливе обмеження методу: для авторів зі стабільним, стилістично уніфікованим кодом (як JoshuaBloch, що дотримується стандартних патернів Effective Java) додавання крос-зв'язків може погіршувати характеризованість, оскільки індивідуальні патерни іменування у такого автора слабко виражені. У таких випадках доцільно використовувати альтернативні типи ознак.

Принципова перевага запропонованої метрики над агрегованим середнім – у можливості виявити невидимі при усередненні ефекти. Наприклад, якщо у двох серіях середня точність турніру однакова (скажімо, 80%), розподіл моделей може бути принципово різним: у одній серії 5 моделей дають 100%, а 34 – 76%; у другій – усі 39 моделей дають 80%. Для практичних застосувань (вибір моделі, ансамблювання) перша ситуація значно вигідніша. Запропонована метрика M100 дозволяє кількісно розрізняти такі ситуації.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У роботі запропоновано нову кількісну метрику оцінювання інформативності словника ознак для класифікації програмних кодів – кількість моделей у турнірі, що досягають 100% точності класифікації на рівні вікон та на рівні класів. Метрика розглядається як доповнення до традиційного двопоказникового оцінювання (точність найкращої моделі + середня точність турніру) і характеризує не одну окрему модель, а розподіл якості за усім набором архітектур турніру.

Експериментально на датасеті з 12 авторів (10 людей та 2 моделі ШІ) у 4 мовах програмування досліджено 22 серії у трьох сценаріях. Показано, що додавання символьних крос-зв'язків до базового словника ознак збільшує кількість моделей зі 100% точністю на рівні класів у середньому на +3,23 моделі на серію. Найбільший приріст виявлено у внутрішньомовному сценарії (+6,25 моделі) та у задачі розрізнення Людина/ШІ (+4,38 моделі). Запропонована метрика забезпечує більш детальну характеристику інформативності словника ознак, аніж агрегована середня точність, і дозволяє виявляти ефекти, невидимі при простому усередненні.

Перспективи подальших досліджень включають: (1) узагальнення метрики на довільний поріг точності T% (M100 → MT для T < 100%); (2) розширення набору моделей ШІ – додавання Gemini, GitHub Copilot, DeepSeek-Coder; (3) параметричну оптимізацію формування МВД (значення N і T); (4) дослідження стабільності метрики через множинні запуски з різними random seeds; (5) поєднання запропонованої метрики з традиційними агрегованими метриками для побудови комплексної системи оцінювання якості словника ознак.

Література

1. Caliskan-Islam A., Harang R., Liu A., Narayanan A., Voss C., Yamaguchi F., Greenstadt R. De-anonymizing Programmers via Code Stylometry. Proceedings of the 24th USENIX Security Symposium. USENIX Association, 2015. P. 255–270.

2. Burrows S., Uitdenbogerd A. L., Turpin A. Comparing techniques for authorship attribution of source code. Software: Practice and Experience. 2014. Vol. 44(1). P. 1–32. https://doi.org/10.1002/spe.2146

3. Abuhamad M., AbuHmed T., Mohaisen A., Nyang D. Large-scale and language-oblivious code authorship identification. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018. P. 101–114. https://doi.org/10.1145/3243734.3243738

4. Abuhamad M., AbuHmed T., Mohaisen D., Nyang D. Large-Scale and Robust Code Authorship Identification with Deep Feature Learning. *ACM Transactions on Privacy and Security*. 2021. Vol. 24, No. 4. Article 23. P. 1–35. https://doi.org/10.1145/3461666

5. Feng Z., Guo D., Tang D., Duan N., Feng X., Gong M., Shou L., Qin B., Liu T., Jiang D., Zhou M. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. Findings of the Association for Computational Linguistics: EMNLP 2020. ACL, 2020. P. 1536–1547.

6. Mitchell E., Lee Y., Khazatsky A., Manning C. D., Finn C. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. Proceedings of the 40th International Conference on Machine Learning. PMLR, 2023.

7. Pan W. H., Chok M. J., Wong J. L. S., Shin Y. X., Poon Y. S., Yang Z., Chong C. Y., Lo D., Lim M. K. Assessing AI Detectors in Identifying AI-Generated Code: Implications for Education. Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training. ACM, 2024. P. 1–11.

8. Голуб М.С. Формування масиву вхідних даних при класифікації текстів у технології інформаційного моніторингу. Математичні машини і системи. 2018. № 1. С. 59–66.

9. Голуб М.С. Формування масиву чисельних ознак для класифікації україномовних текстів в інформаційній технології інтелектуального моніторингу: дис. канд. техн. наук: 05.13.06. Черкаси, 2018. 137 с.

10. Голуб С.В., Мартинова Г.І., Голуб М.С. Моделювання діалектного тексту в технології багаторівневого інформаційного моніторингу. Математичні машини і системи. 2016. № 4. С. 76–83.

11. Guo D., Ren S., Lu S., Feng Z., Tang D., Liu S., Zhou L., Duan N., Svyatkovskiy A., Fu S., Tufano M., Deng S. K., Clement C., Drain D., Sundaresan N., Yin J., Jiang D., Zhou M. GraphCodeBERT: Pre-training Code Representations with Data Flow. International Conference on Learning Representations (ICLR). 2021.

12. Івахненко О.Г. Індуктивний метод самоорганізації моделей складних систем. Київ: Наукова думка, 1981. 296 с.

References

1. Caliskan-Islam, A., Harang, R., Liu, A., Narayanan, A., Voss, C., Yamaguchi, F., Greenstadt, R. (2015). De-anonymizing Programmers via Code Stylometry. In: Proceedings of the 24th USENIX Security Symposium. USENIX Association, pp. 255–270.

2. Burrows, S., Uitdenbogerd, A. L., Turpin, A. (2014). Comparing techniques for authorship attribution of source code. Software: Practice and Experience, 44(1), 1–32. https://doi.org/10.1002/spe.2146

3. Abuhamad, M., AbuHmed, T., Mohaisen, A., Nyang, D. (2018). Large-scale and language-oblivious code authorship identification. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 101–114.

https://doi.org/10.1145/3243734.3243738

4. Abuhamad, M., AbuHmed, T., Mohaisen, D., Nyang, D. (2021). Large-Scale and Robust Code Authorship Identification with Deep Feature Learning. ACM Transactions on Privacy and Security, 24(4), 23. https://doi.org/10.1145/3461666

5. Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., Zhou, M. (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In: Findings of the Association for Computational Linguistics: EMNLP 2020. ACL, pp. 1536–1547.

6. Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., Finn, C. (2023). DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. In: Proceedings of the 40th International Conference on Machine Learning. PMLR.

7. Pan, W. H., Chok, M. J., Wong, J. L. S., Shin, Y. X., Poon, Y. S., Yang, Z., Chong, C. Y., Lo, D., Lim, M. K. (2024). Assessing AI Detectors in Identifying AI-Generated Code: Implications for Education. In: Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training. ACM, pp. 1–11.

8. Holub, M. S. (2018). Formuvannia masyvu vkhidnykh danykh pry klasyfikatsii tekstiv u tekhnolohii informatsiinoho monitorynhu [Input data array formation in text classification within information monitoring technology]. Matematychni mashyny i systemy [Mathematical Machines and Systems], 1, 59–66 [in Ukrainian].

9. Holub, M. S. (2018). Formuvannia masyvu chyselnykh oznak dlia klasyfikatsii ukrainomovnykh tekstiv v informatsiinii tekhnolohii intelektualnoho monitorynhu [Numerical feature array formation for Ukrainian text classification in intelligent monitoring information technology]: dys. ... kand. tekhn. nauk: 05.13.06. Cherkasy, 137 p. [in Ukrainian].

10. Holub, S. V., Martynova, H. I., Holub, M. S. (2016). Modeliuvannia dialektnoho tekstu v tekhnolohii bahatorivnevoho informatsiinoho monitorynhu [Dialect text modelling in multi-level information monitoring technology]. Matematychni mashyny i systemy [Mathematical Machines and Systems], 4, 76–83 [in Ukrainian].

11. Guo, D., Ren, S., Lu, S., Feng, Z., Tang, D., Liu, S., Zhou, L., Duan, N., Svyatkovskiy, A., Fu, S., Tufano, M., Deng, S. K., Clement, C., Drain, D., Sundaresan, N., Yin, J., Jiang, D., Zhou, M. (2021). GraphCodeBERT: Pre-training Code Representations with Data Flow. International Conference on Learning Representations (ICLR).

12. Ivakhnenko, A. G. (1981). Induktivnyi metod samoorganizatsii modelei slozhnykh sistem [Inductive method of self-organisation of models of complex systems]. Kyiv: Naukova Dumka, 296 p.