https://orcid.org/0000-0002-8698-0972
https://orcid.org/0000-0002-9078-3242
https://orcid.org/0000-0003-1845-5014

УДК 004.58+004.94

В.Г. ІВАНОВ[*], О.І. ЗАХОЖАЙ[*], О.О. КРЯЖИЧ[**]

АДАПТИВНИЙ АЛГОРИТМ ЗАПОБІГАННЯ ПОГРІШНОСТЯМ МАТЕМАТИЧНИХ МІРКУВАНЬ У СИСТЕМАХ ШТУЧНОГО ІНТЕЛЕКТУ

[*]Східноукраїнський національний університет імені Володимира Даля, м. Київ, Україна
[*]Інститут телекомунікацій і глобального інформаційного простору НАН України, м. Київ, Україна

Анотація. *У роботі проведено визначення особливостей вирішення простих арифметичних задач із логічною складовою за допомогою чат-ботів із генеративним штучним інтелектом. Як такий чат-бот розглянуто Copilot. Виявлено, що при поглибленні чи уточненні запиту ймовірність отримання вірної відповіді зменшується, бо розширення параметрів запиту призводить до виникнення погрішностей при формуванні відповіді. Подібна проблема дійсно існує і називається обмеженнями великих мовних моделей у математичних розмірковуваннях. Запит користувача і пошук інформації на питання відбувається нелінійно. Користувачі змінюють, доповнюють запити, що лише погіршує кінцеву відповідь чат-боту. Для мінімізації виникнення подібних погрішностей і неточностей пропонується використовувати підхід з нелінійною Z-апроксимацією. Враховуючи, що Z-перетворення базуються на адаптивних алгоритмах та здатні змінювати структурні особливості цих алгоритмів, то при кожній ітерації за таким алгоритмом буде відбуватися наближення до деякої точки, що містить вірну відповідь, а не віддалятися за межі області пошуку. У цьому випадку поведінку користувача, який додає до чат-боту з генеративним штучним інтелектом нові обмеження та уточнення до задачі, можна описати через рекурентні співвідношення породжуваних дрібно-раціональних апроксимацій. У підсумку це дозволяє побудувати напрям руху до вірної відповіді з обмеженням несуттєвих для розуміння задачі зауважень. Запропонований алгоритм дозволяє виконати такі обчислення для дій користувача, що описуються основними тригонометричними функціями. Алгоритм апробований для функцій sin x, cos x, tg x, arctg x мовою програмування Python із застосуванням бібліотеки TensorFlow. У підсумку алгоритм дозволив отримувати вірне рішення задачі, проте зафіксовано збільшення часу виконання завдання при ускладненні логічних умов.*

Ключові слова: *велика мовна модель, чат-бот, Copilot, запит користувача, рекурентне співвідношення, апроксимація.*

Abstract. *The paper identifies the features of solving simple arithmetic problems with a logical component using chatbots with generative artificial intelligence. Copilot is considered as such a chatbot. It was found that when the query is extended or clarified, the probability of obtaining the correct answer decreases because expanding the query parameters leads to errors in forming the answer. Such a problem really exists and is called the limitations of large language models in mathematical reasoning. The user's query and the search for information to answer their question occur non-linearly. Users change and supplement their queries, which only worsens the final answer of the chatbot. To minimize the occurrence of such errors and inaccuracies, it is proposed to use an approach with a non-linear Z-approximation. Given that Z-transformations are based on adaptive algorithms and are able to change the structural features of these algorithms, with each iteration of such an algorithm, there will be an approximation to a certain point containing the correct answer and not moving beyond the search area. In this case, the behavior of the user who adds new constraints and refinements to the problem to the chatbot with generative artificial intelligence can be described through the recurrent relations of the generated small-rational approximations. As a result, this allows you to build a direction of movement to the correct answer with a restriction of remarks that are not essential for the essence of understanding the problem. The proposed algorithm enables performing such calculations for user actions that are described by basic trigonometric functions. The algorithm has been tested for the functions sin x, cos x, tg x, arctg x in the Python programming language using the TensorFlow library. As a result, the algorithm allowed us to obtain the correct solution to*

the problem, but an increase in the task execution time was recorded when the logical conditions became more complicated.
Keywords: *large language model, chatbot, Copilot, user query, recurrence relation, approximation.*

DOI: 10.34121/1028-9763-2025-1-42-54

1. Вступ

Активізація використання інструментів із генеративним штучним інтелектом, що працюють на основі великих мовних моделей (LLM), не лише викликала інтерес до їхньої форми представлення відповідей на запитання користувачів, а й порушила питання про виникнення погрішностей при вирішенні найпростіших арифметичних задач [1]. Зокрема, були виявлені проблеми, які деякі дослідники пов'язали з недосконалістю LLM [2]. Пояснювалося це тим, що алгоритми LLM не можуть розпізнавати логічну складову деяких завдань, через що при вирішенні застосовують лише математичні алгоритми, які призводять до вірних за змістом математичних дій, але невірного рішення задачі [1]. Для доведення цього були проведені масштабні дослідження на кількох найсучасніших відкритих і закритих моделях. Було створено шаблонний тест із генеруванням різноманітних питань, що поєднують арифметичні та логічні складові для дослідження міркування моделей. Висновки довели, що LLM демонстрували помітну дисперсію при відповідях на різні варіанти одного й того ж питання. Крім того, була досліджена нестабільність формування відповіді з математичних задач у моделях із погіршенням продуктивності при збільшенні кількості пунктів у запитанні. Тобто в цілому був зроблений висновок про виникнення обмежень у LLM при необхідності справжнього логічного міркування, у зв'язку з чим інструменти з генеративним штучним інтелектом починають відтворювати кроки з простих арифметичних дій за наявними даними без урахування суті задачі. У підсумку це призводить до втрати продуктивності через відсутність гнучкості алгоритму в залежності від змін логічного забарвлення питань.

У зазначеному випадку вирішити проблему можливо через застосування адаптивного алгоритму, тобто алгоритму, зміст якого змінюється в залежності від вхідної інформації [3]. Варто зазначити, що останнім часом адаптивні алгоритми використовувалися для вирішення складних задач, які потребують вибору найкращого рішення [4]. Адаптивне керування з упередженим зв'язком, представлене в роботі [5], використовує алгоритм нейронної мережі для ідентифікації точної моделі системи в режимі онлайн і введення її до рекурсивного обчислення, що може ефективно зменшити системну помилку, спричинену неточністю моделі. У роботі [6] зазначається, що при вирішенні низки складних задач, які потребують розпізнавання послідовностей, вибору ознак, традиційні методи не завжди ефективні та викликають низьку продуктивність обчислень. Зазначається, що для подібних задач традиційні методи більш складні, зі слабкою адаптивною здатністю до послідовностей різної довжини та стійкістю до різних цілей. Тому, безумовно, рішенням є розробка методів розпізнавання послідовності даних або пунктів питань на основі часових згорткових мереж з адаптивним до довжини послідовності алгоритмом.

Дослідження [7] стосовно адаптивних алгоритмів приводять до думки, що жодна окрема алгоритмічна ідея не є достатньо потужною та гнучкою, щоб вирішити обчислювальну проблему. Але будь-який алгоритм може бути оцінений за його найгіршою продуктивністю на будь-яких вхідних даних певного розміру. У книзі наводиться приклад застосування адаптивних алгоритмів для навчання нейронних мереж на основі даних провідних дослідників, які представили різні аспекти цієї галузі з найважливішими моделями і результатами. Саме у цьому дослідженні наголошується на використанні апроксимації для отримання результатів, які ще не мають чіткого за значенням розуміння у нейронній мережі.

Використання апроксимації для алгоритмізації різноманітних нестаціонарних процесів представлено у дослідженнях [8, 9]. І хоча в роботах досліджуються різні прикладні напрями, але загальна думка зводиться до одного: рідкісні процеси потребують окремого дослідження, особливо процеси, які характеризують динаміку, як це відбувається при використанні інструментів із LLM. Найбільший інтерес за темою дослідження мають роботи [10, 11], в яких розглядаються загальні апроксимаційні підходи до вирішення задач управління складними системами.

Тож, не зважаючи на змістовні дослідження, питання побудови адаптивних алгоритмів для різноманітних динамічних процесів залишаються відкритими. Особливо це стосується рішення прикладних задач щодо інструментів штучного інтелекту, зокрема, для запобігання виникненню погрішностей при формуванні відповіді на запит користувача.

Метою роботи є представлення математичних основ формування адаптивних алгоритмів запобігання погрішностям, які виникають при вирішенні у системах з генеративним штучним інтелектом математичних задач із логічною складовою в умові.

Задачі роботи:

– представити рішення мінімізації погрішностей і неточностей за допомогою рекурентних співвідношень при вирішенні задач за допомогою інструментів, що використовують адаптивний штучний інтелект;

– представити адаптивний алгоритм на основі апроксимації функцій, які описують певний процес пошуку рішення математичної задачі на запит користувача.

2. Постановка задачі мінімізації погрішностей за допомогою адаптивного алгоритму з апроксимацією функцій

Досліджуючи особливості формування запиту від користувача до чат-ботів з генеративним штучним інтелектом, можна помітити особливість, коли на запити з логічною складовою отримується помилкова або неточна відповідь. При цьому такий чат-бот, наприклад, як Copilot, пропонує поглибити чи уточнити запит. Але після уточнення можна отримати ще більш віддалену від істини відповідь. Тобто, виникає ситуація, коли розширення параметрів запиту призводить до виникнення погрішностей при формуванні відповіді.

Подібні погрішності, що призводять до виникнення неточної або помилкової відповіді, визначаються у роботі [1] як обмеження LLM у математичних розмірковуваннях. А враховуючи той факт, що зараз іде активне машинне навчання LLM [2] разом із збільшенням доступності інструментів генеративного штучного інтелекту для пересічного користувача, то подібні заявлені обмеження значно погіршують продуктивність роботи [1].

Якщо прийняти, що запит користувача може бути описаний через функцію кута, де запит формується в певному секторі за координатами точок кола, то такий запит може бути представлений тригонометричними функціями. Подібне можна пояснити через розуміння деякої області інформації, якій належить точка, де інформація відповідає на поставлене питання точно, інші точки утримують додаткову або супутню інформацію, а точки, що знаходяться за межами області, несуть суперечливу або неправдиву інформацію. У математичному вираженні зазначене можна виразити через сектор S_l — припустимої області зміни показників джерела інформації, яка представлена колом (сферою) O_l з центром у деякій точці O та радіусом R_l (рис. 1). При цьому l виступає вектором пошуку, спрямованим до точки M_l. Якщо в деякий момент часу t' інформація буде відповідати точці M_l, то вірогідність отримання вірної інформації P_l у секторі S_l, згідно з джерелом інформації, буде оцінюватися, як 1. Наближення інформаційних даних до межі допустимої області O_l буде визначати зменшення вірогідності отримання вірної інформації на запит. На самій межі припустимої області буде дорівнювати 0. Час t_l^o, за тривалості якого $P(S_l,t)=0$

можна назвати критичним за джерелом S_l. Якщо брати до уваги всю сукупність джерел інформації про роботу системи S, то її можна описати деяким критичним часом роботи, за який у систему взаємодії користувача та чат-боту вводиться додаткова інформація за запитом, допоки не виникнуть обмеження, що призводять до невірного формування відповіді:

$$t_{kp} = min\,(t_1^0, t_2^0, ..., t_p^o).$$

Наведений запис дозволяє описати стан системи S в момент часу, коли ймовірність отримання вірної інформації про систему стає мінімальною:

$$S(t) = min\,(P_1(S,t), P_2(S,t)).$$

Введення додаткових уточнень, що розширюють запит, призводить до ситуації, коли пошук здійснюється за більшим колом джерел інформації. При цьому відповідність запиту такої інформації може стати мінімальною, оскільки розширення сутності запиту відбувається хаотично. В результаті цього радіус R_l збільшуватиметься, розширюючи область змін показників. Тобто необхідна інформація вже вишукуватиметься не у конкретному секторі за темою запиту, а за межами кола, що описує предметну область запиту.

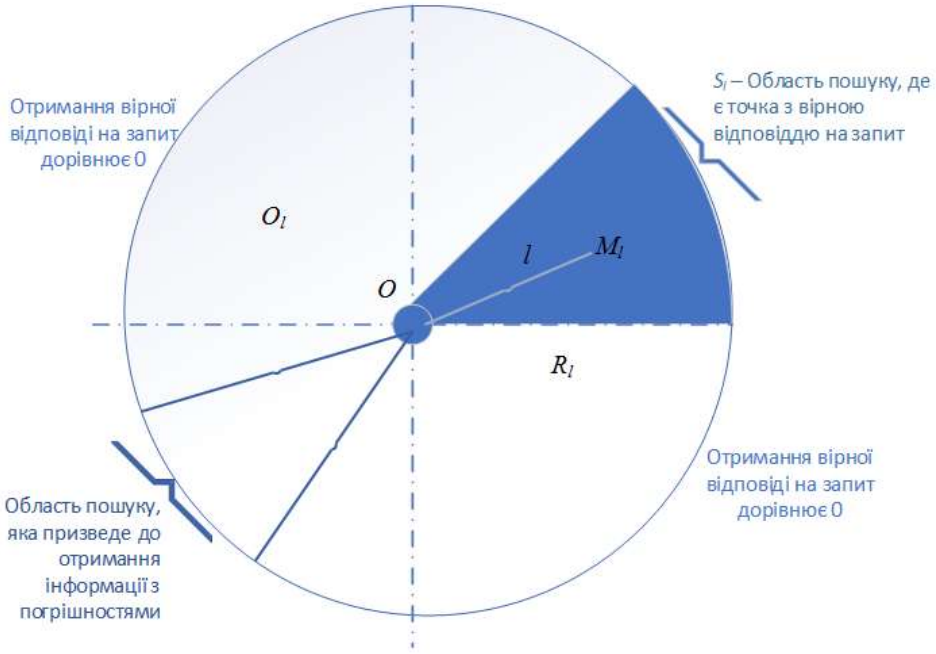

Рисунок 1 — Область зміни показників джерела інформації при пошуку

При формуванні запиту на пошук інформації через чат-боти з генеративним штучним інтелектом, користувач застосовує природну мову, якою описує сектор пошуку. У такому запиті можна визначити ключове слово чи слова, за якими відбувається пошук відповіді, що знаходиться у точці M_l, та уточнення сектору пошуку, тобто обмеження, недотримання яких автоматично присвоює відповіді значення «хиба». Збільшення радіусу R_l із розширенням області показників у такий спосіб призводить до виникнення погрішностей і неточностей при формуванні відповіді користувачу.

Мінімізувати подібні погрішності вбачається можливим із використанням підходу, викладеного в роботі [12], за допомогою нелінійної Z-апроксимації. Враховуючи, що Z-перетворення за [13] базуються на адаптивних алгоритмах та здатні змінювати структурні

особливості цих алгоритмів, то при кожній ітерації за таким алгоритмом буде відбуватися наближення до деякої точки M_l, що містить вірну відповідь, а не віддалятися за межі області пошуку. Використовуючи модель із роботи [12], можна визначити значення двох базових параметрів m та n, які визначатимуть кількість необхідних ітерацій для знаходження вірної відповіді, та фактор, який буде проігнорований у запиті як такий, що не поглиблює сутність запиту:

$$\Delta \approx C_n\left(x/N^m\right)^n \approx \left\lceil x/N^{mn+l}\right\rceil,$$

де C_n — константа, що має залежність від *n*;

N — величина зменшення інтервалу;

x — формалізація запиту, створеного природною мовою.

Апроксимація початкового або заключного наближення вірної відповіді розглядається через рекурентне співвідношення $l = -\log_N |C_n|$.

Користувач, звертаючись до чат-боту з генеративним штучним інтелектом, оперує різними за обсягом даними для запиту, поглиблюючи та спрощуючи запит. Подібну поведінку можна описати тригонометричними функціями на певному проміжку часу для дослідження періодичної події. Формування запиту для вирішення математичної задачі за допомогою інструментів із генеративним штучним інтелектом можна вважати такою подією. Відповідно, запобігання погрішностям міркувань можна здійснити за базовою послідовністю рекурентних співвідношень, розуміється послідовність рекурентних формул.

У підсумку зазначене дозволить створити адаптивний алгоритм, який у залежності від того, якою тригонометричною функцією може бути описана поведінка користувача при формуванні запиту до чат-боту, буде реалізовувати розрахунки кроків до наближення точки M_l зі значенням «істина» з мінімізацією впливу фактора n, який не є суттєвим для запиту, визначеного за ключовим словом та обмеженнями. Деякі подібні розрахунки можуть призвести до збільшення кількості ітерацій та витрат часу на формування відповіді на запит. Подібне можна мінімізувати за допомогою підходу Г.С. Теслера до вирішення задач із розкладанням функцій за нев'язками [11]. У підсумку це дозволить створювати більш гнучкі адаптивні алгоритми, необхідні для машинного навчання LLM.

3. Мінімізація погрішності та неточності за допомогою рекурентних співвідношень

Послідовність рекурентних співвідношень може бути представлена рекурентними формулами, *p*-член яких може бути за n_p більшим, ніж n_{p-1} у виразі $Z(nx_m) = f[Z(x_m)]$, де $x_m = x/n^m$, що описує формалізацію запиту з пошуку інформації при вирішенні математичної задачі.

Найпростіший приклад задачі такий: дитина збирає яблука, в перший і другий день вказана *j*-та кількість яблук, а у третій день зазначено, що яблук зібрано вдвічі більше, ніж у перший день. Але декілька яблук із зібраної партії в половину менші. Приклад подібної задачі та її вирішення за допомогою чат-боту Copilot наведено на рис. 2.

Рисунок 2 — Приклад вирішення задачі за допомогою чат-боту Copilot з наданням відповіді з погрішністю

У наведеному на рис. 2 прикладі погрішність виникає через додаткову логічну складову, де зазначається про зменшення розмірів окремих яблук. Це чат-ботом Copilot сприймається як обмеження, що у підсумку призводить до невірної відповіді. Реалізація зазначених послідовностей рекурентних співвідношень дозволить формалізувати запит, вирішуючи задачу за окремими кроками.

Припустимо, що поведінка користувача при формуванні запиту описується синусоїдою, тобто, як у наведеному прикладі, до задачі додаються суттєві і несуттєві обмеження, які враховуються чи не враховуються при вирішенні задачі тільки за допомогою логічних операцій. У цьому випадку

$$Z_{m-1} = \sum_{k=0}^{n} (-1)^k C_{2n+1}^{2k+1} Z_m^{2k+1} (1 - Z_m)^{n-2k},$$

де $Z_m = \sin\left[x/(2n+1)^m\right]$, $m = m_0$, $m_0 - 1$, ..., 0, звідки $Z_0 = \sin x$.

Або у вигляді спрощення:

$$Z_{m-1} = T_{2n+1}(Z_m),$$

де $T_{2n-1}(Z_m)$ — багаточлен Чебишова.

Для обчислення $tg\,x$ можна використати метод Ейлера [15]:

$$Z_{m-1} = \frac{nZ_m}{1-} \frac{(n^2-1)Z_m^2}{3-} \frac{(n^2-4)Z_m^2}{5-\ldots} \frac{(n^2-p)Z_m^2}{-2p+1-\ldots},$$

де $Z_m = tg\dfrac{x}{n^m}$, $Z_0 = tg\,x$.

Подібним чином можна представити рекурентні співвідношення породжуваних дрібно-раціональних апроксимацій за різними напрямками поведінки користувача при пошуку вирішення задачі. Наприклад, відгалуження від основного завдання задачі шляхом пояснення несуттєвих для рішення додаткових умов та обмежень за допомогою $ctg\,x$ (рис. 3), візуалізацію чого можна реалізувати мовою програмування Python:

```
import matplotlib.pyplot as plt
import numpy as np

# Створюємо масив значень x від -2π до 2π
x = np.linspace(-2 * np.pi, 2 * np.pi, 1000)

# Обчислюємо значення косинуса та котангенса для всіх x
y_cos = np.cos(x)
y_cot = 1 / np.tan(x)

# Створюємо графік
plt.figure(figsize=(10, 6))

plt.plot(x, y_cos, label='cos(x)', color='blue')

plt.plot(x, y_cot, label='cot(x)', color='red')
plt.legend()

plt.title
plt.xlabel('x')
plt.ylabel('y')

plt.grid(True)

plt.ylim(-10, 10)
plt.show()
```

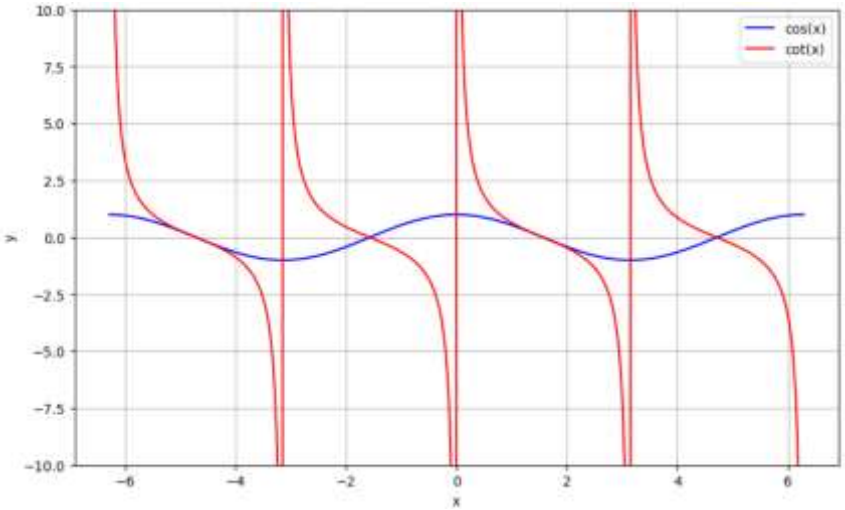

Рисунок 3 — Візуалізація різноманіття поведінки користувача при пошуку рішення через чат-бот

Для мінімізації погрішностей і неточностей через надмірну інформацію при запиті можна використати для обчислення $ctg\,x$:

$$Z_m = \frac{\sum_{k=0}^{[n/2]} (-1)^k C_n^{2k} Z_{m-1}^{n-2k}}{\sum_{k=0}^{(n-1)/2} (-1)^k C_n^{2k+1} Z_{m-1}^{n-2k-1}},$$

де $Z_m = ctg\frac{x}{n^m}$, $Z_0 = ctg\,x$.

Деталізація розмірковувань із застосуванням *Z*-апроксимації для мінімізації помилок наближення наведена у роботі [16]. Застосований у цій роботі підхід передбачає використання кусково-лінійних атракторів при наближенні, що дозволяє знайти нелінійний шлях, який визначає та ігнорує несуттєві обмеження задачі, задовольняє всі суттєві обмеження та формує відповідь на запит користувача з максимально можливою точністю результату. Фактично означене є підходом до створення пошукових алгоритмів у масивах неструктурованої інформації за допомогою чат-ботів із породжувальним штучним інтелектом, коли перехід від початкової точки до точки-джерела інформації на інтервалі описується якоюсь тригонометричною функцією, рішенням якої буде знаходження фіксованої точки, за якою може бути знайдена найбільш точна відповідь за поставленим запитом.

4. Створення адаптивного алгоритму на основі Z_m -апроксимації

При вирішенні арифметичної задачі, обтяженої логічними обмеженнями, які призводять до розширення пошуку інформації в масивах неструктурованої або слабо систематизованої інформації, неодмінно будуть накопичуватися похибки трансформування інформації або округлення даних. Саме тому при практичній реалізації методів апроксимації та створення адаптивних алгоритмів основні питання будуть пов’язані з коректністю та стійкістю.

Виходячи із зазначеного вище, адаптивний алгоритм запобігання погрішностям математичних міркувань у системах штучного інтелекту може бути представлений такими кроками:

Крок 1. Введення умов задачі природною мовою користувача.

Це може бути задача аналогічно наведеній на рис. 2. Граматичні помилки природною мовою користувача значення не мають.

Крок 2. Відбувається перевірка задачі.

Граматичні помилки виправляються, задача розбивається на суттєві та несуттєві умови й обмеження. Несуттєві ігноруються. Несуттєвою умовою можна вважати вказівку, що декілька яблук мають менший розмір за інші. Пошук щодо відповідності зазначеного умовам задачі відбувається за окремим розгалуженням алгоритму. Адаптація відбувається за питанням, яке поставлене в задачі. При правильному ігноруванні несуттєвих умов задачі наступний крок приведе до отримання відповіді на запит.

Крок 3. Пошук відповіді.

На цьому етапі може бути отримана така відповідь, як на наведеному рис. 2. У випадку, коли несуттєві умови ігнорувалися, то відповідь буде адекватною і це стане завершенням алгоритму. Однак у разі, якщо відповідь не задовольнила користувача, він може ввести додаткові умови.

Крок 4. Введення користувачем додаткових умов, обмежень, пояснень до задачі.

Повернення до кроків 1–3. Проходження з порівнянням відповідей за першим та другим циклами пошуку. Скільки б додаткових умов не було введено, вони розглядати-

муться за окремими паралельними циклами. Після цього кроку наступні кроки до кроку 7 реалізуються окремо.

Крок 5. Запит користувача з додатковими умовами за кроком 4 розбивається на логічні пари слів за темою запиту.

Формується робоча таблиця розбиття. Виводяться окремо задача, окремо пояснення, обмеження, додаткові фактори. В залежності від того, ускладнив користувач задачу чи полегшив, обробка інформації йде за своїми алгоритмами в залежності від моделі поведінки користувача при взаємодії з чат-ботом.

Крок 6. Перебір логічних пар слів.

З логічних пар слів за перебором формуються різні варіанти запиту. Кожен результат проходить перевірку на відповідність умовам задачі та адекватності відповіді за окремим розгалуженням алгоритму. Перебір відбувається, допоки всі пари слів робочої таблиці не пройдуть можливі варіанти перебору.

Крок 7. Аналіз варіантів відповіді за кожним реалізованим алгоритмом перебору.

Всі отримані відповіді аналізуються на адекватність. Формується нова робоча таблиця, де розміщуються найбільш відповідні відповіді за кожним алгоритмом обробки інформації. Відбувається вибір відповіді, яка задовольняє всі необхідні властивості інформації.

Крок 8. Виведення результату.

Формування відповіді на запит користувача та виведення на екран. Реакція користувача є критерієм, куди ця відповідь буде віднесена у процесі машинного навчання чат-бота: до правильної, на яку треба орієнтуватися, чи до невірної, яку треба запам’ятати і не припускати більше.

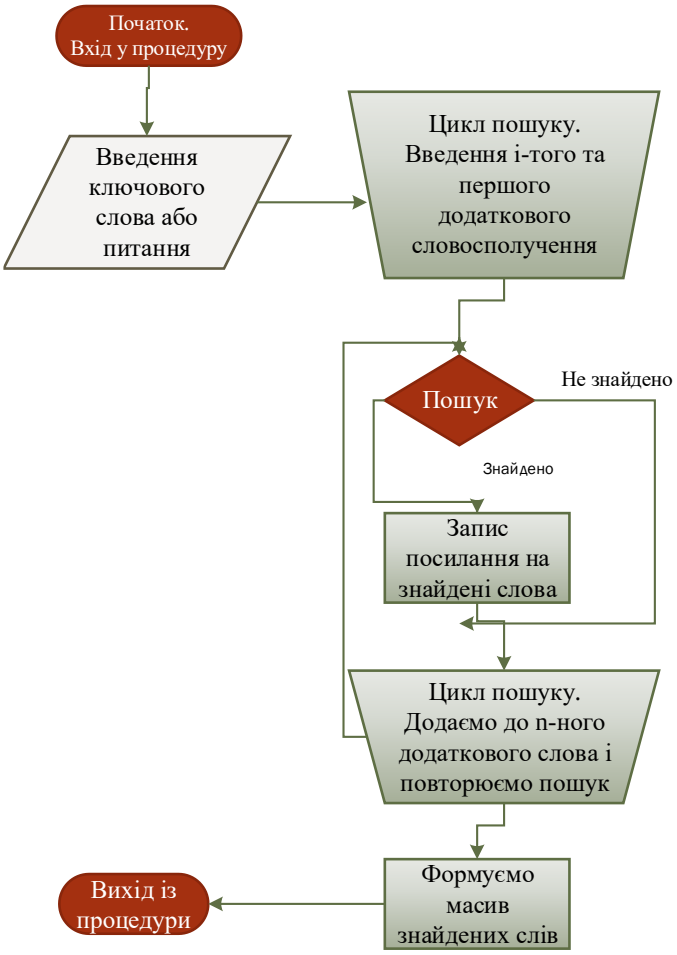

Рисунок 4 — Блок-схема першої процедури виконання адаптаційного алгоритму

Наведений алгоритм можна поділити на окремі частини, які в собі містять обрану кількість дій, необхідних для виконання на певному етапі до здійснення адаптаційного кроку:

– процедура I (дії з 1 по 3), робиться крок адаптації за першим ключовим словом. Блок-схема процедури наведена на рис. 4;

– процедура II (дії 4–6), робиться крок адаптації за першим ключовим словом або поставленим питанням та словами з масиву (рис. 5);

– процедура III (дії 7–8), виконуються кроки з адаптацією за знайденими варіантами відповідей на запит користувача (рис. 6).

Після цього виконуються порівняння та вибірка відповіді.

За ключовим словом, питанням та створеним за алгоритмом (рис. 4) масивом слів здійснюється попарний перебір всіх можливих варіантів: питання — пара слів із масиву. Результати вносяться до робочої таблиці, що приймає участь у реалізації таких процедур.

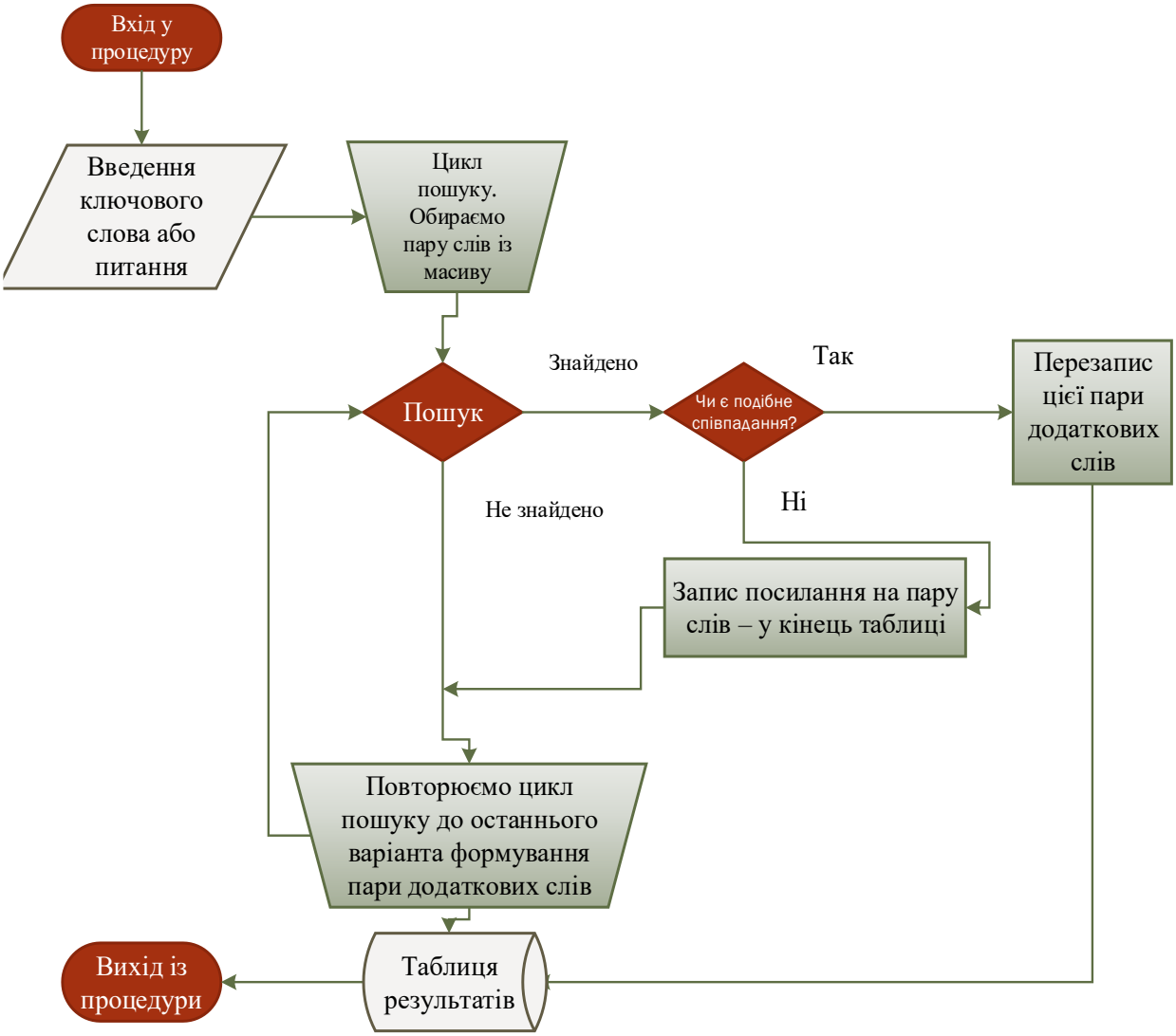

Рисунок 5 — Блок-схема другої процедури виконання адаптаційного алгоритму

При виконанні третьої процедури відбувається об’єднання отриманих варіантів відповідей у нову робочу таблицю з вибором відповіді, яка найбільш точно відповідає на поставлену задачу. Щодо такої відповіді проводиться процедура багатоетапної перевірки, включаючи перевірку з несуттєвими умовами задачі, які були відкинуті при здійсненні першої процедури.

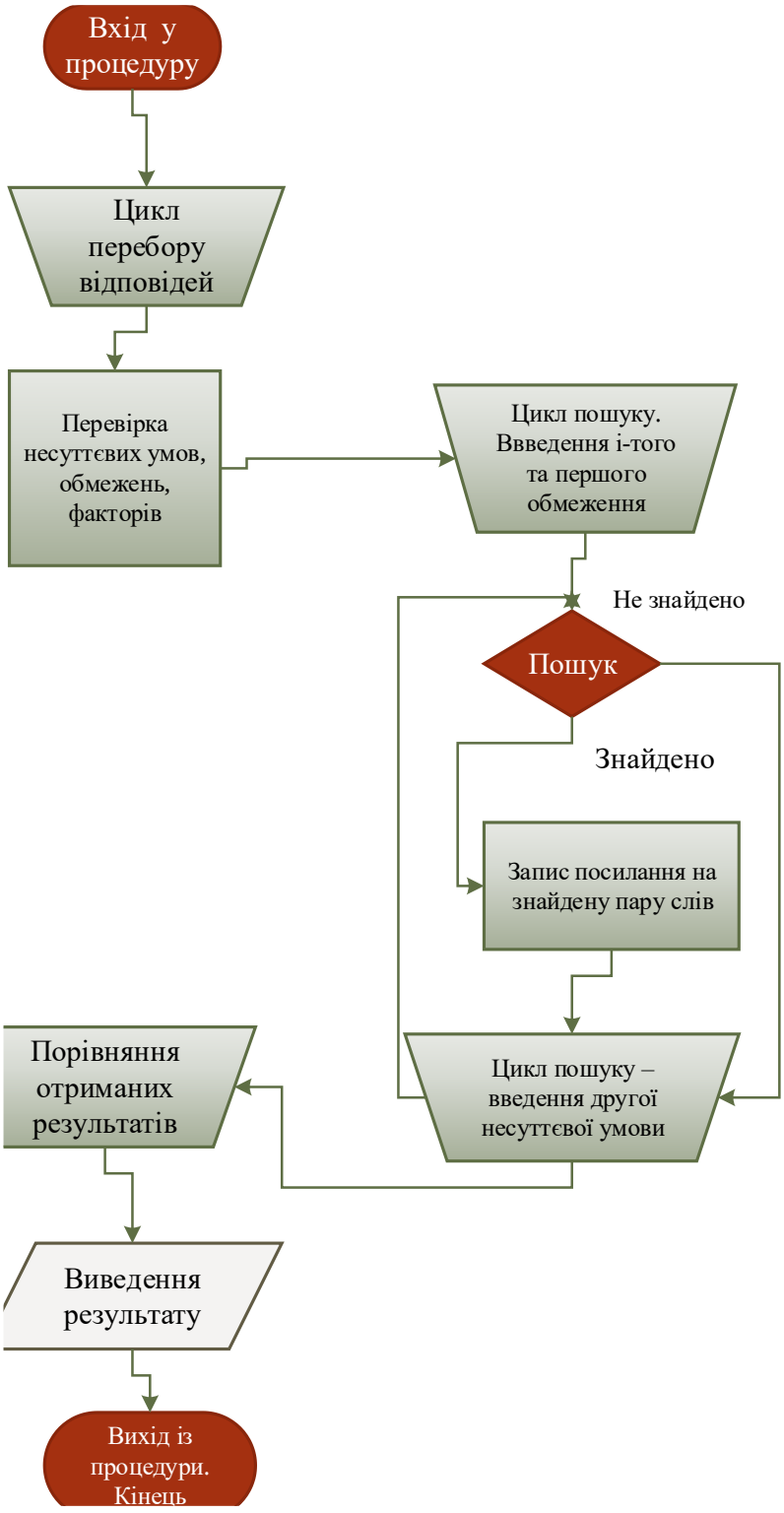

Рисунок 6 — Блок-схема третьої процедури виконання адаптаційного алгоритму

Наведений алгоритм був апробований на базі лабораторії розширеної реальності XR lab та лабораторії Nero Lab Східноукраїнського національного університету імені Володимира Даля шляхом реалізації для функцій *sin x*, *cos x*, *tg x*, *arctg x* мовою програмування Python із застосуванням бібліотеки TensorFlow. Попереднє тестування алгоритму дозволило отримати вірне рішення арифметичної задачі з логічною складовою, аналогічною наведеній на рис. 2. Відмічено збільшення часу обробки інформації для вирішення задачі при

ускладненні логічних умов завдання, що пов’язано зі створенням додаткових розгалужень при виконанні процесів адаптації згідно з введеними додатковими умовами.

6. Висновки

У підсумку проведеної роботи можна зробити такі висновки та узагальнення.

1. Користувач, звертаючись до чат-боту з генеративним штучним інтелектом, оперує різними даними, перемежовуючи їх, що у підсумку призводить до неточної відповіді. Для запобігання виникнення погрішностей та неточностей запропоновано реалізовувати за допомогою рекурентних співвідношень, які дозволяють наближати первинний запит до знаходження вірної відповіді, обмежувати вплив несуттєвих факторів.

2. Розроблений адаптивний алгоритм на основі Z_m-апроксимації для вирішення прикладних задач, що мають логічні складові чи обмеження за умовами. Алгоритм апробований для функцій *sin x, cos x, tg x, arctg x* мовою програмування Python із застосуванням бібліотеки TensorFlow. У підсумку алгоритм дозволив отримувати вірне рішення задачі, проте зафіксовано збільшення часу виконання завдання при ускладненні логічних умов.

Наведені особливості алгоритмізації процесів мінімізації похибок апроксимації при вирішенні прикладних задач можуть бути використані при розробці алгоритмів машинного навчання чат-ботів та інших інструментів на основі генеративного штучного інтелекту, що працюють на основі LLM із метою покращення процесів математичних міркувань для вирішення задач із логічною складовою.

СПИСОК ДЖЕРЕЛ

1. Mirzadeh I., Alizadeh K., Shahrokhi H., Tuzel O., Bengio S., Farajtabar M. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. 2024. P. 2–22. DOI: https://doi.org/10.48550/arXiv.2410.05229.
2. Zhao W., Zhou K., Junyi L., Tianyi T., Wang X., Hou Y., Min Yi., Zhang B., Zhang J., Dong Z., Du Y., Yang C., Chen Y., Jiang J., Ren R., Li Y., Tang X., Liu Z., Wen Ji-R. A Survey of Large Language Models. 2023. DOI: https://doi.org/10.48550/arXiv.2303.18223.
3. Zhao H., Cao Z., Chen Y. Robust Total Least Mean M-Estimate Normalized Subband Filter Adaptive Algorithm Under EIV Model in Impulsive Noise. *Circuits, Systems, and Signal Proc*. 2024. P. 1–27. DOI: https://doi.org/10.1007/s00034-024-02841-9.
4. Lv S., Zhao H., Xu W. Robust Variable Step Size Widely Linear Complex-Valued Least Mean M-estimate Adaptive Algorithm: Derivation and Performance Analysis. *Circuits, Systems, and Signal Processing*. 2024. Vol. 43 (6). P. 3888–3908. DOI: https://doi.org/10.1007/s00034-024-02637-x.
5. Wang M., Fang X., Wang Y., Ding J., Sun Y., Luo J., Pu H. A dual-loop active vibration control technology with an RBF-RLS adaptive algorithm. *Mechanical Systems and Signal Processing*. 2023. Vol. 191. DOI: https://doi.org/10.1016/j.ymssp.2022.110079.
6. Wang X., Wang P., Song Y., Xiang Q., Li J. Recognition of high-resolution range profile sequence based on TCN with sequence length-adaptive algorithm and elastic net regularization. *Expert Systems with Applications*. 2024. Vol. 248. DOI: https://doi.org/10.1016/j.eswa.2024.123417.
7. Roughgarden T. Beyond the worst-case analysis of algorithms. Cambridge University Press, 2021. 706 p. ISBN: 9781108637435. DOI: https://doi.org/10.1017/9781108637435.
8. Dörfler D. On the Approximation of Unbounded Convex Sets by Polyhedra. *Journal of Optimization Theory and Applications*. 2022. Vol. 194 (1). P. 265–287. DOI: https://doi.org/10.1007/s10957-022-02020-3.
9. Xu J., Wang J., Rao J., Zhong Y., Zhao S. Twin Deterministic Policy Gradient Adaptive Dynamic Programming for Optimal Control of Affine Nonlinear Discrete-time Systems. *International Journal of Control, Automation and Systems*. 2022. Vol. 20 (9). P. 3098–3109. DOI: https://doi.org/10.1007/s12555-021-0473-6.
10. Теслер Г.С. Обобщенные адаптивные аппроксимации функций. *Математичні машини і системи.* 1998. № 2. С. 3–8.

11. Теслер Г.С. Адаптивные аппроксимации и итеративные процессы. *Математичні машини і системи*. 2004. № 2. С. 22–41.
12. Кряжич О.О., Трофимчук О.М. Метод обробки неструктурованої інформації на вебресурсах/ *Проблеми керування та інформатики*. 2022. № 67 (4). С. 106–115. DOI: https://doi.org/10.34229/2786-6505-2022-4-8.
13. Rabiner L.R., Gold B., Yuen C.K. Theory and Application of Digital Signal Processing. *IEEE Transactions on Systems, Man and Cybernetics*. 1978. Vol. 8, N 2. P. 146–146. DOI: https://doi.org/10.1109/TSMC.1978.4309918. URL: https://ieeexplore.ieee.org/document/4309918.
14. Mickens R.E. Generalized trigonometric and hyperbolic functions. CRC Press, 2019. 212 p. eBook ISBN 9780429446238. DOI: https://doi.org/10.1201/9780429446238.
15. Butcher J.C. Numerical Methods for Ordinary Differential Equations. New York: John Wiley&Sons, 2003. 484 p. ISBN 978-0-471-96758-3. URL: https://learn.fmi.uni-sofia.bg/pluginfile.php/99883mod_resource/content/1/%21%21%21%21%5BButcher_J.%5D_Numerical_methods_for_ordinary_differ%28BookZZ.org%29.pdf.
16. Kryazhych O., Kovalenko O. Examining a mathematical apparatus of *Z*-approximation of functions for the construction of an adaptive algorithm. *Eastern-European Journal of Enterprise Technologies*. 2019. Vol. 3 (99). P. 6–13. DOI: https://doi.org/10.15587/1729-4061.2019.170824.

Стаття надійшла до редакції 24.01.2025