УДК 004.8

П. В. Здебський; А. Ю. Берко, д-р техн. наук, проф.

ПЕРЕВІРКА ТЕКСТУ ПІСЛЯ ГЕНЕРАЦІЇ ЗА ДОПОМОГОЮ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ДЛЯ ФІЛЬТРУВАННЯ НЕПРАВИЛЬНИХ ВІДПОВІДЕЙ

Сьогодні особливо актуальною є задача узгодження великих мовних моделей. Моделі настільки здатні, що можуть вирішити багато різних завдань, використовуючи підхід zero-shot. Але оскільки вони стали більш здатними, вони знаходять обхідні шляхи для вирішення завдань не так, як очікують дослідники. Це особливо небезпечно у виробничому середовищі, оскільки важко контролювати вихід моделі, яка була навчена бути універсальною. У цій роботі пропонується використовувати одну і ту ж модель кілька разів в різній формі з метою покращення якості згенерованого тексту.

Подальший розвиток отримав метод підвищення точності моделей генерації текстового контенту. Це дозволяє користувачеві не наводити десятки прикладів бажаної та небажаної поведінки моделі, оскільки сама модель може робити це автоматично. Тобто, на відміну від звичайних методів підвищення точності моделі, які вимагають навчального набору моделей, запропонований підхід включає етап ідентифікації. В результаті ідентифікації ми отримуємо набір прикладів, на яких модель автоматично навчається і тим самим підвищує свою точність.

У цій роботі було запропоновано два конкретні методи. Перший метод просто використовує модель дискримінатора для перевірки результатів моделі генератора та запитує повторно створити текст, якщо результати не відповідають критеріям користувача. За допомогою цього підходу вдалося видалити всі неправильні генерації, але за рахунок позначення третини правильних як неправильні. Другий підхід більш складний, він окрім дискримінатора також використовує модель імітатора. Процес вимагає, щоб модель імітатора генерувала зразки введених користувачем даних, потім генератор генерував текст відповіді для кожного зразка, після чого дискримінатор перевіряв згенеровані результати та додавав їх до навчальних даних. Це підвищило точність з 56 % до 66 % у задачі логічного висновку.

Ключові слова: *gpt-4, задача узгодження, генерація тексту, обробка природної мови, задача логічного висновку.*

Вступ

Із збільшенням розміру моделей опрацювання природної мови більш очевидною стає проблему їх узгодження. Задача узгодження є однією із підзадач безпеки штучного інтелекту [1, 2]. У цій роботі запропоновано методи багатоетапного генерування текстів з використанням великих мовних моделей для узгодження згенерованого тексту із запитом користувача.

Для покращення точності великих мовних моделей використовують різні стратегії. Розробник GPT-4, компанія OpenAI, вказує розбиття складних задач на простіші як одну із стратегій зміни вхідного тексту (prompt engineering) для покращення результатів. Наприклад одна модель класифікує запит користувача, а інша пропонує вирішення його проблеми. Ще одна стратегія що пропонується це дати час на роздуми. Наприклад, модель витягнула із книги всі абзаци що стосуються певної теми, але після цього можна її запитати щоб вона перевірила чи ніякі абзаци вона не забула.

Сьогодні активно ведуться дослідження, що пов'язані із багатоетапною генерацією тексту [3]. Прикладом може бути робота Large Language Models are Zero-Shot Reasoners у якій запропоновано запитати модель описати кроки для вирішення задачі [4]. Після того як вона згенерує кроки, їх разом із оригінальною задачею передають цій моделі на вхід щоб вона дала остаточну відповідь. У роботі Bootstrapping Reasoning With Reasoning також запропоновано генерувати покрокове вирішення задачі, а потім додавати його до тренувального набору даних [5]. Додаватись будуть лише ті покрокові вирішення які ведуть до правильної відповіді, правильність можна перевірити, бо працюємо із розміченими

даними. Після доповнення тренувального набору новими даними ми ще раз проходимось по прикладах, для яких було неправильно згенеровано покрокове вирішення.

Ще одним прикладом може бути робота Self-Consistency Improves Chain of Thought Reasoning in Language Models де модель генерує багато відповідей на одне питання, а кінцевою відповіддю вважається та, яка найчастіше зустрічається [6].

У роботі Training Verifiers to Solve Math Word Problems тренується модель верифікатор, який буде перевіряти генератор [7]. Схожий підхід було використано і у цій роботі, але запропонований підхід не вимагає розмічених даних.

Мета роботи і постановка задачі

У цій роботі для покращення якості згенерованого тексту використовується багатоетапний підхід, що включає модель дискримінатор для перевірки результатів генерації. Тобто після того як основна модель згенерує текст, він передаються на вхід іншій моделі, яка від основної відрізняється вхідним текстом інструкцією (prompt). Ця інша модель дискримінатор визначає чи відповідає результат оригінальному запиту користувача.

Розглядається два способи використання дискримінатора, щоб дослідити чи повторне передавання на вхід великій мовній моделі згенерованих прикладів може покращити якість генерування. Перший для відсіювання неправильно згенерованого тексту і другий для розмітки результатів генерації для збільшення тренувальних даних основної моделі. У другому на вхід передаються приклади у яких модель допустила помилку, але вказується, що це приклади негативної поведінки, для того, щоб модель більше не допускала схожих помилок. Процес є повністю автоматичним, бо розміткою займається дискримінатор.

Програмна реалізація

Користувач просить модель згенерувати текст, який буде відповідати певним правилам, і надає кілька прикладів. Модель використовує введені користувачем дані для генерування відповіді, але перед її відображенням модель перевіряє, чи згенерований текст відповідає правилам, що задані користувачем. Якщо ні, він додається як зразок негативної поведінки для дотреновування моделі. Щоб перевірити цей підхід, було обрано просте завдання генерувати імплікацію з речення. Архітектуру системи представлено у таблиці 1.

Таблиця 1

Компоненти системи

Назва компонента архітектури системи	Призначення
get_baseline_model_response	Отримання результатів базової моделі GPT-4.
get_classification_model_response	Класифікація результатів базової моделі на істинні та хибні (хибні результати використовуються для покращення моделі).
get_trained_model_response	Отримання результатів покращеної моделі.

Алгоритм функціонування моделі включає такі базові кроки:

- Генерація тексту на основі базової моделі.
- Ідентифікація відповідності критеріям користувача.
- Тренування фінальної моделі, що враховує негативні приклади поведінки з попереднього кроку.
- Генерація тексту на основі фінальної моделі.

Після проведення початкового експерименту було вирішено змінити підхід. У попередньому підході набір висловлювань ділився на дві групи, але було вирішено виділити окремий набір даних для тестування всіх моделей, бо в такому випадку ми імітуємо

поведінку моделі в реальних умовах використання. Тепер модель буде самостійно генерувати собі приклади запитів користувача. Через те, що у попередньому підході ми давали додатковий набір даних моделі порівняно з базовою, то якість могла покращуватись через більшу кількість даних [8].

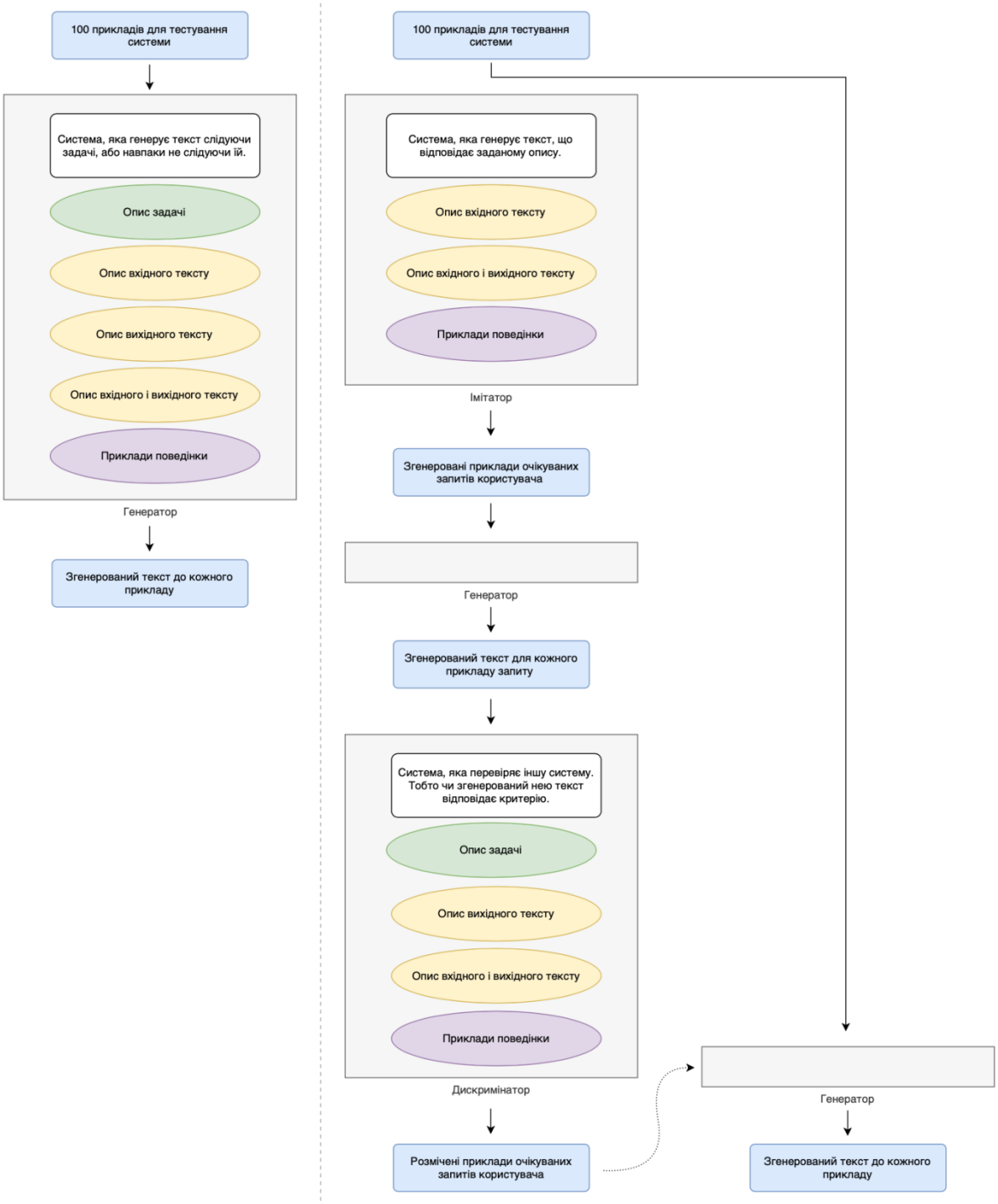

Рис. 1. Використанням імітатора та дискримінатора для збільшення тренувальних даних

Запропоновано два методи покращення текстових генеративних моделей. Перший це коли після генерації модель-дискримінатор відкидає текст, якщо він не відповідає оригінальному запиту користувача. Текст генерується декілька разів поки дискримінатор не підтвердить, що текст правильний. А другий полягає у тому, що модель-імітатор генерує певну кількість потенційних запитів користувача. Потім до кожного запиту генерується відповідь. Після цього кожну відповідь перевіряє модель-дискримінатор і з відповідною міткою передає його

у тренувальний набір даних. Модель із цим розширеним тренувальним набором даних генерує текст за оригінальним запитом користувача.

Для того щоб використовувати запропонований підхід потрібно сформулювати задачу, описати вимоги до вхідного і вихідного тексту, надати декілька прикладів правильної і неправильної поведінки системи. Перший метод дає значне покращення результатів і може бути використаний для будь-якої задачі пов'язаної з генерацією тексту. Другий має більш вузьке застосування. Особливо коли немає ресурсів підготувати для моделі більше декількох прикладів і якщо легше сформулювати задачу показуючи неправильні приклади.

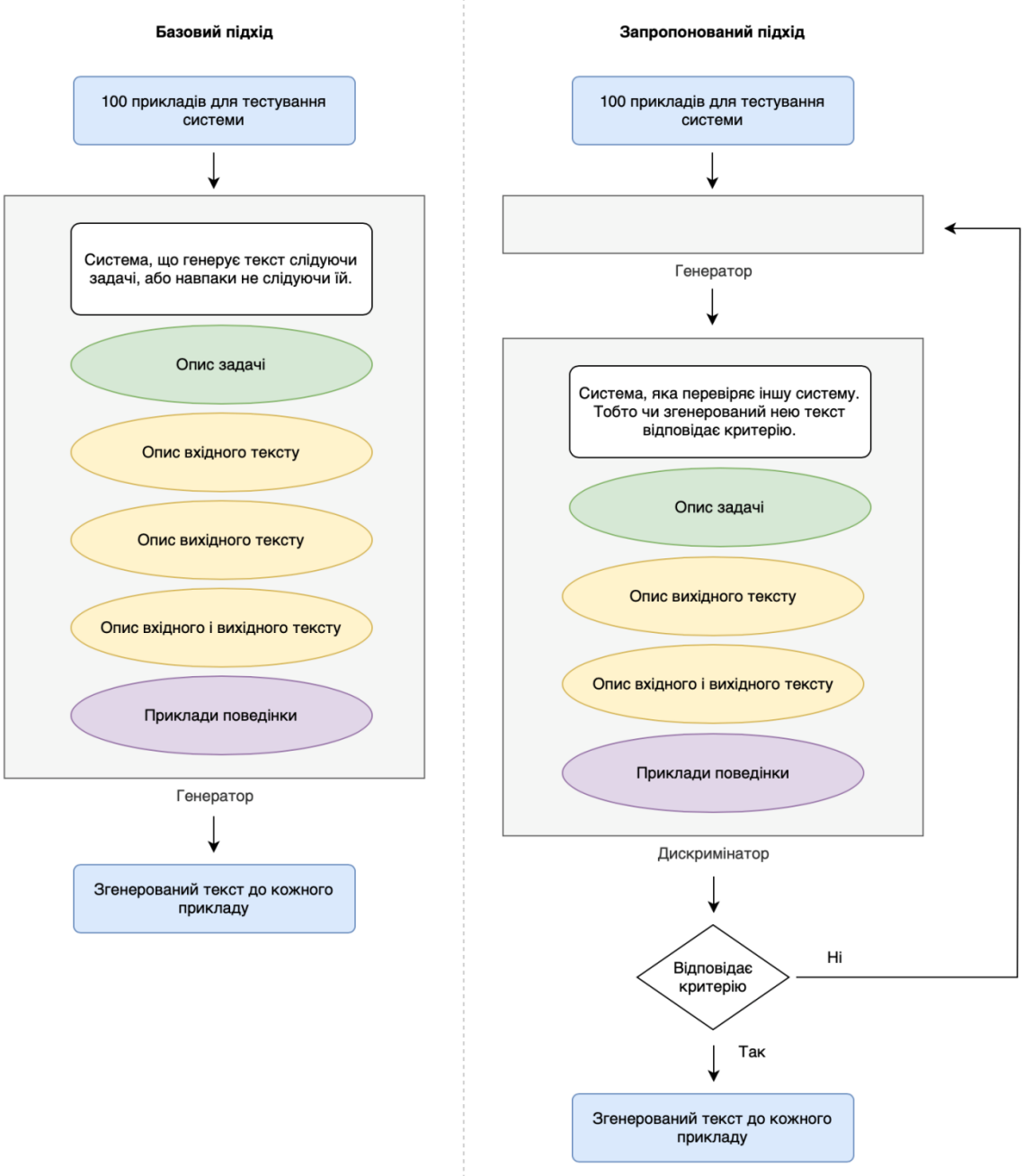

Рис. 2. Метод із використанням дискримінатора для перевірки генерації

Запропоновано метод генерування тексту з використанням верифікатора для оцінки результатів після того, як основна модель згенерує текст. Це дало можливість покращити точність генерування тексту на прикладі задачі логічного висновку [9].

Загальний опис методу:

1. Користувач робить запит x.

2. Модель G генерує N прикладів. За замовчуванням $N = 40$.
3. Модель I класифікує кожен із N прикладів на відповідність запиту користувача x.
4. Приклади z, що не відповідають запиту користувача x, зберігаються.
5. Створюється копія моделі G, якій передають приклади z. Копія G із прикладами негативної поведінки z утворює модель T, що враховує ці приклади для генерації, на відміну від оригінальної моделі G.
6. Модель T використовується для відповіді на оригінальний запит користувача x.

Аналіз результатів

У підході із дотреновуванням моделі на нових прикладах точність зросла із 56 % до 66 %. Метрика точності рахувалась як відношення правильних генерацій до всіх генерацій на 100 прикладах. Є дві причини, що обмежують приріст точності. Це мультиплікація помилки та те, що збільшення кількості прикладів недостатньо значне. Але більше прикладів, більший вплив помилок у тренувальних даних і більша вартість, що вимагає тонке настроювання (fine-tuning, який є обмежений у GPT-4). Варто звернути увагу, що запропоновані підходи тестувались на задачі логічного висновку (NLI). Прикладом є запит "Об'єкт має всі кути 90 градусів", базова модель відповіла "Об'єкт є квадратом", а запропонована "Об'єкт є прямокутником".

Підхід із моделлю-дискримінатором для перевірки згенерованого тексту це простіший підхід, який також тестувався на 100 прикладах з тією ж метрикою. Результат тестування це 60 правильних класифікацій "так", 27 правильних класифікацій "ні", 0 неправильних класифікацій "так", та 13 неправильних класифікацій "ні". Тобто модель-дискримінатор правильно відкине близько 40 % відповідей згенерованих системою, бо класифікує їх як ті, що не відповідають критерію користувача. Із тих, що модель відкинула, близько двох третіх буде відкинуто правильно. Тобто модель дискримінатор не пропустила жодної неправильної генерації, але з тих що позначила неправильними одна третя були правильними генераціями. Прикладом пари речень, які відсіяла модель дискримінатор є запит "Об'єкт вивчає фізику" і відповідь базової моделі "Об'єкт є студентом", бо не тільки студенти вивчають фізику.

Приклад необроблених результатів базової моделі показано на зображенні нижче. Кожен рядок — це окрема пара тверджень і мітка, яка вказує, чи є зв'язок між ними. Горизонтальна лінія «|» є роздільником. Перше висловлювання у кожному рядку визначається користувачем, другий є висловлюванням, яке на думку моделі, випливає з першого. Після цього до моделі-дискримінатора робиться запит, щоб визначити чи є логічний зв'язок між парою речень, тому мітка також генерується моделлю. Пари тверджень потім додаються до тренувального набору даних остаточної моделі.

```
Об'єкт має крила. | Об'єкт може літати. | ні
Об'єкт має здатність мислити або обробляти інформацію. | Об'єкт є істотою або комп'ютером. | так
Об'єкт містить англійські слова. | Об'єкт може бути використаний для вивчення англійської мови. | так
Об'єкт може бути використаний для вивчення англійської мови. | Об'єкт містить англійські слова або фрази. | так
Об'єкт є твариною. | Об'єкт є живим істотою. | так
Об'єкт може плавати у воді. | Об'єкт не втопиться у воді. | так
Об'єкт здатен говорити людською мовою. | Об'єкт є розумним істотою. | так
Об'єкт є раритетним автомобілем. | Об'єкт має високу колекційну вартість. | так
Об'єкт експонується у музеї. | Об'єкт є предметом мистецтва або історії. | так
Об'єкт покритий шерстю. | Об'єкт може допомогти зігрітися в холодну погоду. | ні
Об'єкт складається з металу. | Об'єкт проводить електрику. | так
Об'єкт має шість сторін. | Об'єкт є кубом. | так
Об'єкт знає правила дорожнього руху. | Об'єкт має водійський дозвіл. | ні
Об'єкт містить англійські слова або фрази. | Об'єкт може бути використаний для вивчення англійської мови. | так
Об'єкт має високу колекційну вартість. | Об'єкт може бути предметом аукціону. | так
Об'єкт проводить електрику.  | Об'єкт може бути визначений як провідник. | так
Об'єкт може показувати погоду. | Об'єкт є метеостанцією. | ні
Об'єкт – це щоденник. | В об'єкті можна робити записи. | так
Об'єкт розміщений у кінотеатрі. | Об'єкт може бути пов'язаний з просуванням фільмів. | так
Об'єкт живе в пустелі. | Об'єкт може витримати високі температури. | так
Об'єкт складається з снігу. | Об'єкт може розтанути при підвищенні температури. | так
Об'єкт не згорить при високих температурах. | Об'єкт є вогнетривким. | так
Об'єкт має доступ до Google Play Store. | Об'єкт є пристроєм на базі Android. | так
Об'єкт є китом. | Об'єкт є морським ссавцем. | так
```

Рис. 3. Приклад результатів дискримінатора

Висновки

Загалом результати показують запропоновані підходи покращують якість генерації тексту. Простіший підхід із перевіркою згенерованого тексту і повторній генерації дає більш суттєві покращення якості генерації. Другий підхід також показує кращі результати ніж базова модель і буде особливо корисним у випадках, коли користувач не може надати моделі достатньо прикладів, щоб описати бажану поведінку моделі для генерації тексту.

СПИСОК ЛІТЕРАТУРИ

1. The alignment problem from a deep learning perspective [Electronic resource] / R. Ngo, L. Chan, S. Mindermann // arXiv. – 2022. – Access mode : https://arxiv.org/abs/2209.00626.

2. Goal Misgeneralization in Deep Reinforcement Learning / L. Langosco; J. Koch, L. D. Sharkey [et al.] // Proceedings of the 39th International Conference on Machine Learning. International Conference on Machine Learning. PMLR. – 2022. – P. 12004 – 12019.

3. Training language models to follow instructions with human feedback [Electronic resource] / L. Ouyang, J. Wu, X. Jiang [et al.] // arXiv. – 2022. – Access mode : https://arxiv.org/abs/2203.02155.

4. Large Language Models are Zero-Shot Reasoners [Electronic resource] / T. Kojima, S. S. Gu, M. Reid [et al.] // arXiv. – 2022. – Access mode : https://arxiv.org/abs/2205.11916.

5. STaR: Bootstrapping reasoning with reasoning [Electronic resource] / E. Zelikman, Y. Wu, J. Mu, N. D. Goodman [et al.] // arXiv. – 2022. – Access mode : https://arxiv.org/abs/2203.14465.

6. Self-consistency improves chain of thought reasoning in language models [Electronic resource] / X. Wang, J. Wei, D. Schuurmans, [et al.] // arXiv. – 2022. – Access mode : https://arxiv.org/abs/2203.11171.

7. Training verifiers to solve math word problems [Electronic resource] / K. Cobbe, V. Kosaraju, B. Mohammad [et al.] // arXiv. – 2021. – Access mode : https://arxiv.org/abs/2110.14168.

8. Intelligent System for Semantically Similar Sentences Identification and Generation Based on Machine Learning Methods / P. Zdebskyi, V. Lytvyn, Y. Burov [et al.] // CEUR workshop proceedings – 2023. – Vol. 2604. – P. 317 – 346.

9. Explaining simple natural language inference / A.-L. Kalouli, A. Buis, L. Real, M. Palmer, V. de Paiva [et al.] // Proceedings of the 13th Linguistic Annotation Workshop. – 2019. – P. 132 – 143.

Стаття надійшла до редакції 01.03.2024.
Стаття пройшла рецензування 18.03.2024.

Здебський Петро Васильович – аспірант кафедри інформаційних систем та мереж, e-mail: petrozd@gmail.com.
Національний університет "Львівська політехніка".

Берко Андрій Юліанович – д-р техн. наук, професор кафедри інформаційних систем та мереж.
Національний університет "Львівська політехніка".