УДК 004.89

ВДОСКОНАЛЕННЯ АГЕНТІВ LLM ШЛЯХОМ ВИКОРИСТАННЯ ЇХ В ЯКОСТІ ПОШУКОВИХ АГЕНТІВ

Стрельцов О.С.
e-mail: oleksii.streltsov@nure.ua
Науковий керівник – к.т.н., доц. Вітько О.В.
Харківський національний університет радіоелектроніки, каф. ШІ
м. Харків, Україна

This work is devoted to increasing the performance and efficiency of large language models, models that learn the function of producing coherent completion of any given natural language text due to their ability to learn from massive amounts of example data, through test-time scaling. Such models gain many emergent capabilities that make them useful as assistants for professionals, but they also lack some common capabilities of classic systems like search-based agents. We compare common search-based agents to decoder-only large language models to find possible solutions to this problem through additional computation.

Сучасні великі мовні моделі здатні виконувати складні задачі з високою точністю, завдяки чому їх часто обирають в якості центрального елементу автоматизації інтелектуальних задач [1]. Проте, такі моделі часто не здатні достатньо ефективно працювати, коли взаємодія з середовищем є тривалою, потребує тренування або реакції на відгуки середовища чи користувача. Такі недоліки відсутні або менш виражені в агентів, що користуються пошуком в просторі станів. Але мовні моделі погано виконують характерні операції, наприклад, планування або бектрекінг [2]. Ми припускаємо, що процес виведення великої мовної моделі та пошуку в просторі станів мають певну схожість, тож їх можна порівняти й проаналізувати, як саме можна вплинути на процес виведення, щоб отримати бажані властивості пошукових алгоритмів. Для порівняння було обрано клас алгоритмів пошуку для задач задоволення обмежень [3] та пошук A* [4].

Алгоритми пошуку для задач задоволення обмежень ведуть пошук не послідовності дій у середовищі, що веде до бажаного стану, а власне самого цільового стану через ефективний перебір змінних за наданими обмеженнями. Основою алгоритмів пошуку для задач задоволення обмежень є послідовне зв’язування змінних, перевірка узгодженості набору змінних та відв’язування змінних у разі конфлікту. В процесі виведення велика мовна модель також обробляє вхідні інструкції та за ними формує певні обмеження, проте, надані природною мовою такі інструкції часто можуть містити неточності або не наводити певні обмеження явно. Відповідно, успішність виведення залежить від

можливості моделі правильно інтерпретувати наведені обмеження і вгадувати неявні. Причому операція перевірки також виконується моделлю неявно, через що результат пошуку фактично є спекулятивним. А операція відв'язування змінних значно ускладнюється через особливості бектрекінгу в мовних моделях.

Алгоритм A* покладається на наявність евристики, що оцінює стан відносно кількості дій, необхідних для переходу в цільовий стан. На основі цієї евристики формується черга із пріоритетом – алгоритм відвідує лише найкращий з доступних станів, але може виконати бектрекінг у випадку, якщо евристика схибила. Для роботи алгоритму необхідно, щоб евристика не занижувала відстань до цільового стану. Якщо порівняти це із запропонованою моделлю виведення-пошуку, то стане очевидним, що у випадку, коли діалог може бути продовжений користувачем, отримати евристику, яка не недооцінює – неможливо. Проблематичним є і бектрекінг за чергою, адже бектрекінг мовних моделей працює за принципом рефлексії, а не повернення до попереднього стану.

З урахуванням наведених порівнянь можна сформувати наступні загальні напрямки покращення методів масштабування під час виконання:

− використання нових генерацій із доповненим контекстом замість довгих діалогів, розділення стану виведення та стану пошуку;

− атрибуція дій до отриманих результатів і застосування отриманого контексту для запобігання повторенню поганих рішень;

− активний вплив на вхідні дані.

Для моделей типу «чорна скриня» пропонуються рішення з залученням моделі до переписування інструкцій та власного аналізу результатів. Rewrite-MCTS – у разі невдачі модель переписує попередній варіант інструкції з використанням відгуку. Операції переписування зберігаються у деревовидній моделі, за допомогою якої обирається найкраща інструкція для переписування. LLM-CSP – виконання моделлю аналізу власної генерації з метою виділення змінних та обмежень задачі, побудова множини конфліктів та її використання для збагачення оригінальної інструкції.

Для моделей типу «біла скриня» пропонуються рішення на основі визначення ключових етапів пошуку через ентропію активацій моделі [5]. Керуючись евристикою, за якою вважається, що високі значення ентропії відповідають стану з кількома значущими альтернативами, можна окремо зберігати значення активацій та згенеровані моделлю токени, використовуючи їх як стани і дії пошуку. Для ефективної роботи зі станами виконується онлайн-кластеризація, де кожному кластеру відповідає певний контекст.

Найпростішим варіантом пошуку на основі такого контексту може бути пошук «грубою силою», за яким на вже використані токени накладається штраф. Як більш витончений варіант пропонується побудова

графа виведення моделі, за яким можна обирати пріоритетні дії в стилі MCTS, а саме відповідно до успішності та популярності шляхів і оцінки самої моделі. Через використання таким методом механізму штрафування окремих дій пропонується використання окремої дії «дослідження», що відповідає множині невикористаних токенів.

У випадку, коли між оцінками рішення можна встановити відношення часткового порядку (наприклад, відсоток пройдених тестів для задач з програмування), можна додатково покращити якість пошуку через введення ймовірності прямого відношення окремих рішень до порушення обмежень задачі.

Для цього достатньо ввести в оцінку стану додатковий множник, що наближається до нуля при значному звуженні розподілу оцінок із зростанням кількості спостережень.

Запропоновані методи демонструють високу ефективність при роботі з великими мовними моделями для локального використання (від 1 до 27 мільярдів параметрів). Rewrite-MCTS вирішує більшу кількість задач, ніж інші сучасні методи, при низькому використанні токенів. LLM-CSP є складним для менших моделей, але добре масштабується з кількістю параметрів.

Методи пошуку на основі ентропії демонструють значну перевагу над звичайним семплюванням і легко поєднуються із іншими методами.

Список використаних джерел:

1. Hybrid Intelligence for Industry 4.0: Integrating Large Language Models and Reinforcement Learning / V. Terziyan та ін. International Journal of Computer Information Systems and Industrial Management Applications. 2025. Т. 17. С. 610–627. DOI: https://doi.org/10.70917/ijcisim-2025-0038.

2. Bachmann G., Nagarajan V. The pitfalls of next-token prediction. URL: https://arxiv.org/abs/2403.06963 (дата звернення: 11.03.2026).

3. Prosser P. HYBRID ALGORITHMS FOR THE CONSTRAINT SATISFACTION PROBLEM. Computational Intelligence. 1993. Т. 9, № 3. С. 268–299. DOI: https://doi.org/10.1111/j.1467-8640.1993.tb00310.x.

4. Hart P., Nilsson N., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics. 1968. Т. 4, № 2. С. 100–107. DOI: https://doi.org/10.1109/tssc.1968.300136.

5. Reasoning with Exploration: An Entropy Perspective / D. Cheng та ін. URL: https://arxiv.org/abs/2506.14758 (дата звернення: 11.03.2026).