https://doi.org/10.31891/2307-5732-2025-349-65
УДК 004.773.2

ЦАП ВЛАДИСЛАВ
Національний університет «Львівська політехніка»
https://orcid.org/0000-0002-8062-0079
e-mail: vladyslav.b.tsap@lpnu.ua

ЗАСТОСУВАННЯ ГРАФОВИХ БАЗ ДАНИХ ДЛЯ ОПТИМІЗАЦІЇ МОВНОЇ МОДЕЛІ

У даній науковій роботі досліджується використання графових баз даних для оптимізації великих мовних моделей шляхом застосування властивостей графових структур. Такий підхід забезпечує низку переваг, зокрема покращення пояснюваності моделі, ефективніше використання контексту для кожного користувача та гнучке коригування вагових параметрів завдяки атомарності підграфів. Особлива увага приділена динамічній адаптації ваг моделі, що дає змогу підвищити її персоналізацію та точність відповіді. Крім того, у роботі розглядається можливість контролю версій та редагування ваг підграфів для оптимізації результатів.

У межах дослідження проведено експериментальні розрахунки, які оцінюють швидкість обробки запитів, обсяг необхідного дискового простору та вимоги до оперативної пам'яті для коректної роботи великої мовної моделі LLaMA 3. Отримані результати порівнянні з традиційними методами зберігання та обробки даних.

Таким чином, дана робота детально досліджує перспективи та виклики використання графових баз даних у контексті оптимізації великих мовних моделей, пропонуючи нові підходи до їхньої масштабованості, ефективності та адаптивності.

Ключові слова: графові бази даних, великі мовні моделі, оптимізація мовних моделей.

TSAP VLADYSLAV
Lviv Polytechnic National University

APPLICATION OF GRAPH DATABASES FOR LANGUAGE MODEL OPTIMIZATION

This research paper investigates the application of graph databases in optimizing large language models, focusing on their ability to enhance interpretability, flexibility, and adaptability. By utilizing the inherent properties of graph structures, the study explores how these databases can improve model explainability through the ability to trace the paths of influence, offering clearer insights into model decision-making processes. Another key benefit is the dynamic adjustment of model weights, which provides flexibility for task-specific needs, such as medical diagnostics or personalized recommendations, leading to more accurate outputs.

The study also examines the operational challenges of using graph databases, including slower query processing speeds compared to in-memory computing, significant storage demands, and the complexities of data management. The volume of data required for large language models further exacerbates these issues, raising concerns about the efficiency of storage and access mechanisms. Despite these challenges, the research highlights the potential of graph databases for specific tasks where scalability and speed are less critical, suggesting that they may be best suited for niche applications rather than broader, general-purpose use.

Experimental assessments conducted on the LLaMA 3 model provide valuable insights into the performance of graph databases, including measurements of query processing speed, memory usage, and disk storage requirements. These results are benchmarked against traditional data storage and processing methods, offering a detailed comparison of efficiency and feasibility. Overall, this study offers a comprehensive analysis of the advantages, limitations, and future prospects of using graph databases for large language model optimization, advocating for a balanced approach that addresses both their potential and practical constraints in real-world applications.

Keywords: graph databases, large language models, language model optimization.

Вступ

Стрімкий розвиток нейронних мереж широко відкриває сферу їхнього застосування. Такі технології все глибше проникають у життя людей, тому що вони допомагають вирішувати повсякденні задачі. Завдяки розробникам, які створили зручні інтерфейси для взаємодії, користувачі мають приємний користувацький досвід. Однак ми розглянемо саме недоліки та пошукаємо методи вирішення для них.

Насамперед, варто наголосити, що великі мовні моделі є трансформерами. Тобто всередині них відбуваються математичні операції, що в результаті дають відповідь, яка з точки зору людини є логічна. Однак сама модель не має такого поняття як «логіка», оскільки вона не може мислити, а є лише результатом математичних операцій на основі тренувальних даних [1].

У разі використання штучного інтелекту у сфері медицини чи права, нам важливо, щоб можна було зрозуміти на основі чого модель надала відповідь, тобто не тільки отримати обґрунтування від самої моделі, а ще мати змогу розібратися у цьому за допомогою вихідного коду. На жаль, останнє не є доступно через саму будову нейронної мережі.

Також можливість редагування вибіркового кластеру ваг також пригодилася, оскільки знаючи за що вони відповідають, ми могли б змінити їхню важливість у контексті всієї моделі.

Окрема проблема може виникнути під час навчання моделі в випадку, якби ми хотіли мати декілька знімків її ваг. Якщо їх експортувати для моделі розміром 70 мільярдів, то ми отримаємо файл розміром в 140 ГБ [2]. Якщо таких знімків нам треба робити декілька, тоді проблема вільного місця і його вартості стає гостро.

У всіх випадках ми можемо застосувати графові бази даних для того, щоб вирішити наші задачі. Даний варіант є перспективний, оскільки відношення між нейроннами можна перенести як відношення між вершинами, а тому ми спробуємо вирішити описані виклики за допомогою графових баз даних.

Аналіз літературних джерел

У першій статті [3] розглядається проблема перетворення природної мови у запити до графових баз даних. Автори підкреслюють проблему відсутності наборів даних та пропонують підхід, який ефективно вирішує дану проблему. Дане дослідження можна використати у моїй роботі, щоб зрозуміти специфіку роботи з графовими базами даних.

У другій статті [4] представлена архітектурна зміна для графових баз даних, завдяки якій автори демонструють масштабування для обробки величезних робочих навантажень у розподілених системах. Для моєї статті це дослідження цінне тим, що демонструє підхід, завдяки якому можна спроєктувати графову базу даних під високі вимоги мовних моделей.

Третя стаття [5] описує інтеграцію мовних моделей з графовими базами даних як базою знань. Автори вказують на проблеми мовних моделей у роботі з фактами, та пропонують вирішення даної проблеми. Цю роботу я використаю, щоб детальніше зрозуміти проблематику точності моделей та способів як можна у ручному режимі її покращити.

Автор четвертої статті [6] пропонує альтернативний підхід до виконання операцій глибокого навчання, використовуючи лише реляційні бази даних. Ця публікація є корисною для мого дослідження, тому що у ній впроваджено покращений підхід до представлення та обробки мовних структур у базах даних, що можна використати для планування структури графової бази даних.

У п’ятій статті [7] автори вимірюють швидкість виконання запитів у графових базах даних та пропонують декілька підходів для їхньої оптимізації. У цій роботі основна увага зосереджена на відношенні «багато до багатьох», що вимагає ретельного аналізу для мого дослідження, оскільки застосування пропонованих оптимізаційних алгоритмів може бути ефективним лише за певних умов.

У процесі аналізу літературних джерел було відкинуто джерела, які виявилися тематично вузькими, базувалися на застарілих даних, або виявилися фрагментарними, без належного теоретичного обґрунтування. Таким чином, у даному розділі враховані лише ті джерела, які є релевантними до моєї роботи та є актуальними на сьогоднішній день.

Методи та моделі

Розглянемо спочатку метод застосування графової бази даних для великої мовної моделі. Оскільки нейрони у мережі можна охарактеризувати як вершини у графі, а ваги — як ребра, тоді ми можемо встановити наступний зв’язок між нейронами:

$$(Neuron\_A) - [WEIGHT\ \{value: 0.73\}] -> (Neuron\_B)$$

Таким чином наша модель буде зберігати всі ваги у графовій базі даних. І цей підхід дає нам декілька переваг у нашій розробці.

По-перше, зв’язки між параметрами тепер моделюються явно. У свою чергу це дозволяє нам покращити розуміння моделі, оскільки ми можемо відстежити причини рішення моделі, завдяки проходженню по шляхам графа. Відповідно для кожної відповіді ми можемо отримати підграф вершин та ребер, які безпосередньо вплинули на неї. Тобто у результаті наша мовна модель стала більш придатною для пояснень та аналізу.

По-друге, можливість аналізувати підграф, який має найбільшу вагу на результат, дає нам змогу знаходити викривлений ваговий розподіл. Наприклад, це може проявлятися тоді, коли тренувальні дані мають ухил в специфічну сторону, тобто набір даних має певне упередження. Більше того, наявність ваг у графовій базі даних дає нам можливість коригувати викривлення такого характеру, щоб отримувати точніший результат на виході. Ми можемо навіть у ручному режимі видаляти вершини чи цілі підграфи, що мають негативний вплив на модель.

По-третє, можливість редагувати ваги у будь-який момент дозволяє нам спроєктувати динамічну архітектуру, коли модель може використати підграф лише для певної задачі, а опісля «від’єднатися» від такого підграфу. Наприклад, це може бути корисно, коли ми хочемо, щоб наша модель виконала медичну діагностику, і лише тоді певний підграф «медицина» активується. Такі підграфи зручні тим, що ми можемо теж редагувати їх в залежності від ситуації. Наприклад, відкрили несумісність двох медичних препаратів, і ми можемо зразу оновити ваги між цими двома нейронами. Перевага динамічності може також бути спроєктована для персоніфікації мережі під користувача: ми заморожуємо більшість ваг у графі, однак дозволяємо певній підмножині динамічно адаптуватися під конкретного користувача. Таким чином наша модель буде видавати контекстуалізовані результати, без необхідності додаткового донавчання моделі.

По-четверте, оперування вагами у графовій базі даних дозволяє нам бачити кожну зміну ваг окремо, через що ми можемо сформувати історію змін, і використати це як змінок моделі. Тобто таким чином ми уникнули повного копіювання всіх ваг, а залишили лише кожне модифіковане ребро в історії змін.

Результати експерименту

Незважаючи на описані переваги, у зазначеному підходу є великий недолік — швидкодія. Мовна модель тримає всі ваги в оперативній пам’яті, завдяки чому генерація відбувається так швидко. На противагу, бази даних тримають свої дані на диску, і будь-які операції будуть в рази повільніші. Окрім витрати на доступ до файлового сховища, бази даних витрачають час на гарантування принципів

довговічності, які також дещо уповільнюють роботу. Єдиною перевагою графових баз даних перед оперативною пам'яттю є лише у задачах при пошуку шляху.

Проведемо заміри швидкодії між операціями у графових базах даних і операціями в оперативній пам'яті та висвітлимо результати у таблиці 1.

Таблиця 1

Заміри швидкодії між операціями у різних середовищах

Операція	Графова база даних	Оперативна пам'ять
Читання	1-10 мсек	0.001-0.1 мсек
Запис	5-50 мсек	0.01-1 мсек
Пошук шляху	10-100 мсек	10-500 мсек

Проаналізувавши результат, можемо зробити висновок, що проведення операцій над графовою базою даних буде від 100 до 1000 разів повільніше, аніж над операціями в оперативній пам'яті. Для більшості випадків, втрата такого масштабу буде критичною і вагомою.

Вагомим недоліком варто сказати, що окрім звичних операцій доступу до елементів, важливо сказати про математичні операції, які оптимізовані для оперативної пам'яті завдяки GPU/TPU. У випадку з графовими базами даних, це буде відбуватися сильно довше.

Інша проблема, яка виникає — це необхідний дисковий простір для даної графової бази даних. Для розуміння масштабів порахуємо значення для графової бази даних Neo4j та великої мовної моделі Llama 3 на 70 мільярдів параметрів, яку ми вже роглядали раніше. Одна вершина 15 байтів, а ребро — 34 байти.

$$\text{Кількість нейронів} = \text{Кількість шарів} \times \text{Нейронів на одному шарі}$$

У Llama 3 70B є 80 шарів. Шари уваги (Self-Attention) виконують перетворення за допомогою ваги уваги, не додаючи нових нейронів, тоді як шари прямого поширення тимчасово збільшують розмірність до чотирикратного значення, а потім повертаються до початкового розміру. Сумарна ефективна розмірність параметрів для одного шару, враховуючи як увагу, так і шари прямого поширення, становить

$$(1 + 4) \times 8192 = 40{,}960 \text{ нейронів}$$

Сумарна кількість нейронів у всій нейронній мережі дорівнює:

$$80 \times 40{,}960 = 3{,}276{,}800 \text{ нейронів}$$

Дізнаємося скільки місця займе зберігання лише нейронів, без їхніх властивостей:

$$3{,}276{,}800 \text{ нейронів} \times 15 \text{ байт/нейрон} = 49.15 \times 10^6 \text{ байтів} = 46.87 \text{ МБ}$$

Цей розмір є невеликим та прийнятним.

Наступним кроком порахуємо скільки місця потрібно для зберігання ребер. Кількість ребер буде відповідати кількості параметрів у моделі, тобто 70 мільярдів. Порахуємо місце, яке необхідне для зберігання такої кількості ребер:

$$70 \times 10^9 \text{ ребер} \times 34 \text{ байт/ребро} = 2.38 \times 10^{12} \text{ байтів} = 2.2 \text{ ТБ}$$

Цей розмір є надзвичайно великим для моделі таких розмірів. Для спрощення розрахунків ми пропустили розрахунок властивостей чи пам'ять, яка зарезервована під потреби бази даних. Також аналізована модель на 70 мільярдів параметрів є відносно невеликою, оскільки флагманські моделі використовують 405 мільярдів параметрів, і тоді розмір ребер за приблизними обрахунками становитиме 12.5 ТБ, що є надзвичайно великим обсягом — тоді базу необхідно буде масштабувати та розподіляти, а це додаткові втрати на швидкодії.

Окрім вартості сховища та його організацію, важливо звернути увагу на швидкість запису. Навіть якщо ми будемо записувати зі швидкістю 100 тисяч операцій на секунду, що є вже викликом для сучасних структур, то нам треба буде приблизно 3 години на повне копіювання ребер у базу даних. Це надзвичайно повільно порівняно із триманням в оперативній пам'яті.

Останнім важливим недоліком варто зазначити, що дана задача зовсім не є тривіальною, тому що для неї наразі немає готових рішень, і послідуюча розробка потребуватиме багато архітектурних змін, щоб модель запрацювала та видавала стабільний результат.

Висновок

Використання графової бази даних разом із великою мовною моделлю є нетривіальним, однак перспективним рішенням. Це має переваги у вигляді покращення пояснюваності моделі, завдяки відстеженню задіяних нейронів, більшій динамічності моделі завдяки атомарності підграфів, та ефективного застосування контексту для кожного з користувачів. Окрім цього, можливість контролю версії допомагає у випадках, де це дійсно потрібно.

Однак задачі, які вирішує дана архітектура є неспівставні через свої обмеження — до 100 разів повільніше генерація тексту, великий об'єм даних, накладні витрати на розподілення бази даних, неоптимізовані математичні операції. Ця ідея може бути корисною у вузьких колах, однак вона настільки

радикально відходить від традиційних архітектур глибокого навчання, що описані проблеми можна вирішити іншими інструментами.

Дана робота вичерпно розкриває тему застосування графових баз даних у поєднанні з великими мовними моделями та вказує на можливі переваги та недоліки даного рішення.

Література

1. Bhattacharjee, A., Moraffah, R., Garland, J., & Liu, H. (2024). *Towards LLM-guided Causal Explainability for Black-box Text Classifiers* (arXiv:2309.13340). arXiv. https://doi.org/10.48550/arXiv.2309.13340

2. Moar, C. (2024). *Compressing Language Models using Low-Rank Decomposition and Characterizing the Accuracy—Efficiency Trade-offs*. https://escholarship.org/uc/item/0t6967h4

3. Liang, Y., Tan, K., Xie, T., Tao, W., Wang, S., Lan, Y., & Qian, W. (2024). *Aligning Large Language Models to a Domain-specific Graph Database for NL2GQL* (arXiv:2402.16567). arXiv. https://doi.org/10.48550/arXiv.2402.16567

4. Besta, M., Gerstenberger, R., Fischer, M., Podstawski, M., Blach, N., Egeli, B., Mitenkov, G., Chlapek, W., Michalewicz, M., Niewiadomski, H., Müller, J., & Hoefler, T. (2023). The Graph Database Interface: Scaling Online Transactional and Analytical Graph Workloads to Hundreds of Thousands of Cores. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–18. https://doi.org/10.1145/3581784.3607068

5. Yang, L., Chen, H., Li, Z., Ding, X., & Wu, X. (2024). Give us the Facts: Enhancing Large Language Models With Knowledge Graphs for Fact-Aware Language Modeling. *IEEE Transactions on Knowledge and Data Engineering*, *36*(7), 3091–3110. IEEE Transactions on Knowledge and Data Engineering. https://doi.org/10.1109/TKDE.2024.3360454

6. Du, L. (2020). *In-Machine-Learning Database: Reimagining Deep Learning with Old-School SQL* (arXiv:2004.05366). arXiv. https://doi.org/10.48550/arXiv.2004.05366

7. Kalumin, H., & Deshpande, A. (2024). *Optimizing Queries with Many-to-Many Joins* (arXiv:2412.16323). arXiv. https://doi.org/10.48550/arXiv.2412.16323

References

1. Bhattacharjee, A., Moraffah, R., Garland, J., & Liu, H. (2024). *Towards LLM-guided Causal Explainability for Black-box Text Classifiers* (arXiv:2309.13340). arXiv. https://doi.org/10.48550/arXiv.2309.13340

2. Moar, C. (2024). *Compressing Language Models using Low-Rank Decomposition and Characterizing the Accuracy—Efficiency Trade-offs*. https://escholarship.org/uc/item/0t6967h4

3. Liang, Y., Tan, K., Xie, T., Tao, W., Wang, S., Lan, Y., & Qian, W. (2024). *Aligning Large Language Models to a Domain-specific Graph Database for NL2GQL* (arXiv:2402.16567). arXiv. https://doi.org/10.48550/arXiv.2402.16567

4. Besta, M., Gerstenberger, R., Fischer, M., Podstawski, M., Blach, N., Egeli, B., Mitenkov, G., Chlapek, W., Michalewicz, M., Niewiadomski, H., Müller, J., & Hoefler, T. (2023). The Graph Database Interface: Scaling Online Transactional and Analytical Graph Workloads to Hundreds of Thousands of Cores. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–18. https://doi.org/10.1145/3581784.3607068

5. Yang, L., Chen, H., Li, Z., Ding, X., & Wu, X. (2024). Give us the Facts: Enhancing Large Language Models With Knowledge Graphs for Fact-Aware Language Modeling. *IEEE Transactions on Knowledge and Data Engineering*, *36*(7), 3091–3110. IEEE Transactions on Knowledge and Data Engineering. https://doi.org/10.1109/TKDE.2024.3360454

6. Du, L. (2020). *In-Machine-Learning Database: Reimagining Deep Learning with Old-School SQL* (arXiv:2004.05366). arXiv. https://doi.org/10.48550/arXiv.2004.05366

7. Kalumin, H., & Deshpande, A. (2024). *Optimizing Queries with Many-to-Many Joins* (arXiv:2412.16323). arXiv. https://doi.org/10.48550/arXiv.2412.16323