Література:

1. Zaidi A. et al. 5G Physical Layer: Principles, Models and Technology Component. Academic Press, 2018. 312 p.
2. Saad Asif 5G Mobile Communications. CRC Press, 2018. 336 p.
3. Harri Holma; Antti Toskala LTE for UMTS: Evolution to LTE-Advanced. Wiley Telecom, 2011. 576 p.
4. Susinder R Gulasekaran and Sundar G Sankaran Wi-Fi 6 Protocol and Network. Artech House Publishers, 2020. 248 p.
5. Вакарчук А. О., Склярук А. В., Верємій Є. В. Оцінка параметрів антен базової станції мобільної мережі радіодоступу на енергетику лінії. *VII Міжнародна науково-практична конференція «Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах»*, м. Чернівці, 8–10 листопада 2018 року. С. 85–86.

DOI https://doi.org/10.36059/978-966-397-479-8-132

ВИКОРИСТАННЯ ШІ У РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА АРХІТЕКТУРИ

Проценко М. М.
здобувач вищої освіти третього (аспірантського) рівня
за спеціальністю 123 – Комп'ютерна інженерія
Міжнародний гуманітарний університет
Науковий керівник: ***Мірошник М. А.***
кандидат технічних наук, професор
Міжнародний гуманітарний університет
м. Одеса, Україна

У сучасному цифровому світі програмне забезпечення стає дедалі складнішим, а вимоги до нього постійно зростають. Традиційні методи розробки не завжди можуть впоратися з цими викликами. Штучний інтелект, зокрема великі мовні моделі (LLM), відкривають нові горизонти в розробці програмного забезпечення та архітектурних рішень [1]. Великі мовні моделі революціонізують підхід розробників та архітекторів до своєї роботи. Вони навчаються на великих наборах даних коду та архітектурних патернів, допомагаючи автоматизувати генерацію коду, виявляти помилки та пропонувати оптимальні дизайнерські рішення. Це не лише прискорює

цикл розробки, але й підвищує загальну якість програмних продуктів, дозволяючи командам відповідати зростаючим вимогам ринку.

Однак існують обмеження традиційних практик оптимізації в архітектурі та кодуванні. Ручні процеси стають неефективними зі зростанням розміру та складності систем. Знання розподілені між окремими фахівцями, що створює ризики при їх відсутності чи неможливості співпраці. Тиск швидких релізів не дозволяє проводити глибокий аналіз та оптимізацію. Традиційні методи не встигають за швидкими технологічними змінами, а людський фактор призводить до високої ймовірності помилок та пропуску критичних проблем. Ці виклики підкреслюють необхідність більш просунутих інтелектуальних інструментів, які можуть доповнити людські можливості та йти в ногу з вимогами галузі [4].

Для оцінки великих мовних моделей у генерації коду та архітектури було використано бенчмарк BigCodeBench, який оцінює моделі на основі задач завершення та генерації коду. Колонки бенчмарку пояснюються таким чином: Complete оцінює здатність моделі виконувати завершення коду на основі структурованих докстрингів, тестуючи її можливості у кодуванні. Instruct вимірює здатність моделі генерувати код на основі природних мовних інструкцій, оцінюючи розуміння людських намірів. Average є середнім значенням показників Complete та Instruct, коли обидва доступні. Elo Rating – це система рейтингу продуктивності, яка починається з 1000 і коригується на основі продуктивності в задачах.

T	Model	Complete	Instruct	Average	Elo Rating
🔶	GPT-4-Turbo-2024-04-09	35.1	29.1	32.1	1235
🔶	Qwen2.5-Coder-32B-Instruct	33.8	27.7	30.8	1231
🔶	GPT-4o-2024-08-06	36.5	25	30.8	1227
🔶	DeepSeek-Coder-V2-Instruct (2024-07-24)	33.1	25.7	29.4	1219
🔶	Claude-3.5-Sonnet-20241022	35.1	25.7	30.4	1211
🔶	Claude-3.5-Haiku-20241022	34.5	25.7	30.1	1210
🔶	DeepSeek-V2-Chat (2024-06-28)	32.4	25	28.7	1207
🔶	Claude-3.5-Sonnet-20240620	33.1	25.7	29.4	1204
🔶	Gemini-1.5-Pro-Exp-0827	31.8	27	29.4	1203
🔶	o1-Preview-2024-09-12 (temperature=1)	34.5	23	28.8	1184
🔶	DeepSeek-Coder-V2-Instruct	29.7	24.3	27	1184

Порівнювалися такі моделі: Qwen2.5-Coder-7B-Instruct – відкрита модель з 7,61 мільярдами параметрів, спеціалізована на генерації коду, розумінні та виправленні, підтримує довгі контексти до 128 тисяч токенів. DeepSeek-Coder-V2-Instruct – відкрита модель типу Mixture-of-Experts (MoE) з версіями на 16 та 236 мільярдів параметрів. GPT-4o – закрита модель від OpenAI з приблизно 1,8 трильйонами параметрів, використовує архітектуру трансформера із значними покращеннями та оптимізаціями. o1-preview – оновлена версія GPT-4 з покращеною здатністю до міркування та виконання складних завдань, має доступ до знань до квітня 2023 року. Claude-3.5-Sonnet – закрита модель від Anthropic

з мультимодальними можливостями. Gemini-1.5-Pro – мультимодальна модель від Google з архітектурою Mixture-of-Experts (MoE), оптимізована для ефективної роботи з довгим контекстом до 1 мільйона токенів.

Проте використання лише великих мовних моделей у розробці та архітектурі має обмеження. Моделі можуть не враховувати специфіку проєкту, внутрішні бібліотеки або власні API, що призводить до неточних рекомендацій. Існує ризик генерації неправдивого або нерелевантного коду – так званих галюцинацій. Використання закритих моделей може призвести до витоку конфіденційної інформації. LLM не мають доступу до внутрішньої документації, баз даних або інших корпоративних ресурсів. Є також обмеження в налаштуванні: закритий код моделей не дозволяє їх тонке налаштування під специфічні потреби організації. Ці обмеження підкреслюють необхідність підходу, який би поєднував потужність LLM з доступом до специфічних знань організації, зберігаючи при цьому безпеку даних.

Щоб подолати обмеження великих мовних моделей та покращити процеси архітектури та розробки програмного забезпечення, пропонується інтеграція LLM з технологією Retrieval-Augmented Generation (RAG). Цей підхід поєднує генеративні можливості LLM з контекстуальною релевантністю, яку надає RAG, шляхом інтеграції зовнішніх баз знань. Retrieval-Augmented Generation (RAG) – техніка, яка покращує роботу LLM шляхом інтеграції зовнішніх баз знань [2, 3]. Це дозволяє моделям отримувати доступ до актуальної та специфічної для домену інформації під час генерації коду чи рекомендацій.

Механізм роботи RAG передбачає створення бази знань, де зберігаються внутрішні документи, код, керівництва та інша релевантна інформація. Документи перетворюються у векторні представлення (ембеддінги) за допомогою моделей типу textembedding-gecko. Введені користувачем запити також конвертуються у ембеддінги. За допомогою пошуку подібності знаходяться релевантні документи. Потім LLM генерує відповіді, використовуючи отриману інформацію, що підвищує точність та релевантність.

Переваги нової моделі проєктування та розробки з використанням RAG включають підвищення точності та релевантності, оскільки відповіді підкріплюються реальними, контекстно специфічними даними, що зменшує помилки. Безпека даних покращується, адже конфіденційна інформація зберігається в межах організації, мінімізуючи ризик витоку даних. Гнучкість та налаштування дозволяють організаціям адаптувати базу знань відповідно до своїх потреб, дозволяючи моделі пристосовуватися до специфічних сфер та вимог. Динамічне навчання означає, що модель може інтегрувати останню інформацію без необхідності перенавчання LLM, що сприяє швидкій адаптації до змін.

Практичні приклади впливу RAG на процеси проєктування та розробки демонструють його ефективність. Наприклад, у сценарії використання внутрішніх API без RAG модель не знає про внутрішній API, що призводить до помилок, тоді як з RAG завдяки доступу до документації API генерується коректний код. У проєктуванні з урахуванням політик безпеки без RAG модель може пропонувати рішення, що не відповідають вимогам безпеки, а з RAG інтеграція політик безпеки забезпечує відповідність та оптимізацію. При вирішенні унікальних помилок без RAG LLM не може допомогти з приватними кодами помилок, тоді як з RAG модель надає точні кроки налагодження, використовуючи внутрішню документацію.

Використання великих мовних моделей у поєднанні з пошуком з доповненою генерацією відкриває нові можливості у розробці програмного забезпечення та архітектури. Цей підхід дозволяє подолати обмеження стандартних LLM, забезпечуючи точність, релевантність та безпеку даних. Інтеграція RAG у процеси розробки сприяє підвищенню ефективності, гнучкості та інноваційності, що є ключовими факторами успіху у сучасному технологічному середовищі.

Література:

1. Alammar J., Grootendorst M. Hands-On Large Language Models: Building and Deploying Large Language Models for Production. США : O'Reilly Media, 2024. 425 с.

2. Chapman L. Retrieval Augmented Generation: A Holistic Guide to Getting Started with RAG Language Models and Applications for Beginners. США : Independent Publishing, 2024. 203 с.

3. Zhao P., Zhang H. Retrieval-Augmented Generation for AI-Generated Content: A Survey. arXiv Computer Science, 2024. URL: https://arxiv.org/html/2402.19473v1 (дата звернення: 22.11.2024).

4. Yetistiren B., Ozsoy I., Ayerdem M., Tuzun E. Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. arXiv Computer Science, 2023. URL: https://ar5iv.labs.arxiv.org/html/2304.10778 (дата звернення: 22.11.2024).

5. Nsaif W.S., Salih H.M., Saleh H.H., Al-Nuaim B.T. Conversational Agents: An Exploration into Chatbot Evolution, Architecture, and Important Techniques. Journal of Artificial Intelligence Research, 2024. URL: https://www.researchgate.net/publication/383147827_Conversational_Agents_An_Exploration_into_Chatbot_Evolution_Architecture_and_Important_Techniques (дата звернення: 23.11.2024).