Дементьєв М. О., гр. КІСП-21
Науковий керівник - д.т.н., професор Кушнір Д. О.

Програмний сервіс векторного представлення програмного коду для застосування у роботі з великими мовними моделями

Стан проблеми. Векторне представлення програмного коду набуває все більшого значення в контексті роботи з великими мовними моделями (LLM) та іншими системами штучного інтелекту. Ця технологія дозволяє перетворювати текстові дані програм у числові вектори фіксованої розмірності, що є ключовим для ефективної обробки коду за допомогою LLM [1, 2].

Існує кілька підходів до векторизації коду, включаючи токенізацію та word embeddings, Abstract Syntax Tree (AST) embeddings, graph-based embeddings та transformer-based embeddings [1, 2, 4, 5]. Кожен з цих методів має свої переваги та недоліки, і вибір конкретного підходу залежить від специфіки завдання та доступних обчислювальних ресурсів.

Важливим аспектом роботи з векторними представленнями коду є ефективне зберігання та пошук у великих обсягах даних. Для цього використовуються спеціалізовані векторні бази даних, такі як Faiss або Milvus, які дозволяють швидко знаходити найближчі вектори у багатовимірному просторі [3, 6].

Постановка задачі. Розробити архітектуру програмного сервісу векторного представлення програмного коду для застосування у роботі з великими мовними моделями. Система повинна надавати користувачам можливість завантажувати файли з кодом, автоматично обробляти їх, створювати векторні представлення та зберігати їх у векторній базі даних. Важливою вимогою є підтримка ефективного оновлення даних на основі хеш-суми файлу та наявність модуля для взаємодії LLM з векторною базою даних.

Розв'язання задачі. Розроблено структурну схему програми, що складається з п'яти модулів:

1. Модуль завантаження файлів для прийому та валідації файлів від користувачів.
2. Модуль векторизації коду для перетворення текстового коду у векторні представлення.
3. Модуль зберігання у векторній базі даних для ефективного запису та пошуку векторів.
4. Модуль взаємодії з LLM для забезпечення інтерфейсу роботи з векторною базою даних.
5. Основний модуль для координації роботи всіх інших модулів та взаємодії з користувачем через API.

Система використовує асинхронну архітектуру та розподілені обчислення для забезпечення високої продуктивності. Для оптимізації

використання обчислювальних ресурсів реалізовано механізм оновлення даних на основі хеш-суми файлу.

Алгоритм роботи системи включає:

1. Отримання файлу від користувача через API сервісу.
2. Валідацію файлу на відповідність необхідним критеріям.
3. Обчислення хеш-суми файлу для його унікальної ідентифікації.
4. Перевірку наявності векторного представлення в базі даних.
5. Векторизацію коду (якщо файл новий або змінений).
6. Збереження вектора у векторній базі даних разом з метаданими.
7. Оновлення метаданих для існуючих векторних представлень.

Особливістю розробленої системи є її здатність ефективно обробляти великі обсяги програмного коду, уникаючи повторної векторизації незмінених файлів і забезпечуючи актуальність векторних представлень для змінених файлів.

Модуль взаємодії з LLM реалізовано як REST API, що дозволяє виконувати різні операції з векторними представленнями коду, такі як пошук найближчих сусідів, кластеризація векторів, обчислення подібності між векторами та генерація нового коду на основі існуючих векторних представлень.

Висновки. Запропонована архітектура програмного сервісу дозволяє ефективно створювати та зберігати векторні представлення програмного коду для застосування у роботі з великими мовними моделями. Система демонструє потенціал для застосування в різних сферах, де потрібен аналіз, пошук та генерація коду, включаючи розробку програмного забезпечення, дослідження у галузі штучного інтелекту та автоматизацію програмування.

Подальші дослідження можуть бути спрямовані на вдосконалення методів векторизації коду, оптимізацію алгоритмів пошуку у векторному просторі та розробку спеціалізованих інтерфейсів для різних типів LLM.

Література.

1. Alon, U., et al. (2019). code2vec: Learning Distributed Representations of Code.
2. Feng, Z., et al. (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages.
3. Johnson, J., et al. (2019). Billion-scale similarity search with GPUs.
4. Gu, X., et al. (2018). Deep code search.
5. Chen, Z., & Monperrus, M. (2019). A literature study of embeddings on source code.
6. Milvus Team. (2021). Milvus: A Vector Database for Scalable Similarity Search and AI Applications.