RAG 와 메모리 주입 시스템을 결합한 하이브리드 AI 코드 리뷰 시스템 아키텍처 제안

홍진혁 [1], 도성룡 [2]
[1] 고려사이버대학교 컴퓨터공학부 학부생
[2] 고려사이버대학교 컴퓨터공학부 교수

redmax45@koreacu.ac.kr, sungryongdo@koreacu.ac.kr

Hybrid AI Code Review Architecture: Integrating Retrieval-Augmented Generation with Memory Injection Systems

Jin-Hyeok Hong[1], Sung-Ryong Do[2]
[1]Division of Computer Engineering, Korea Cyber University
[2]Division of Computer Engineering, Korea Cyber University

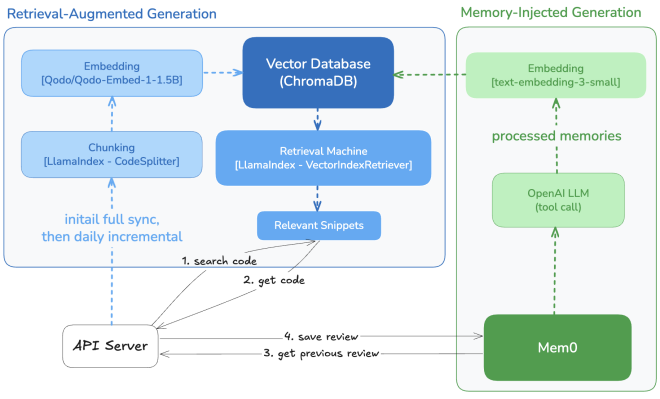
요 약

AI 코드 리뷰 시스템은 개발 효율성과 코드 품질 향상을 위한 핵심 도구이다. 그러나 상용 LLM 기반 접근은 전체 코드베이스 맥락 제공과 리뷰 일관성 측면에서 한계가 있다. 본 연구는 RAG 와 메모리 주입 시스템을 결합한 하이브리드 AI 코드 리뷰 시스템 아키텍처를 제안한다. 제안 시스템은 OpenAI 에이전트가 스스로 Function Calling 을 통해 연관 코드 스니펫을 검색하고 Mem0 를 활용하여 이전 리뷰 내용을 메모리에 저장함으로써 일관성 있는 리뷰를 생성한다. 본 연구는 실용적인 AI 코드 리뷰 시스템 구축을 위한 하이브리드 접근법의 효과성을 파일럿 적용을 통해 검증하였다.

1. 서론

AI 코드 리뷰 시스템은 리뷰어의 코드 리뷰 효율성 향상과 코드 품질 관리를 지원하는 핵심 도구로 주목받고 있다. 하지만 실제 운영 환경에서는 자체 LLM 모델 구축 및 운영 비용이 현실적 제약으로 작용한다. 이에 따라 상용 LLM 서비스를 활용하면서 전체 코드베이스의 맥락을 효과적으로 제공할 수 있는 RAG(Retrieval-Augmented Generation) 기반 접근법이 대안으로 제시되고 있다. RAG 는 외부 데이터 소스에서 관련 정보를 검색하여 LLM 의 답변 생성 과정에 활용함으로써 정확도를 향상시킨다[1]. 효과적인 AI 코드 리뷰는 단순한 오류 검출을 넘어 인간 리뷰어와 같은 자연스러운 피드백을 제공하고 핵심 개선사항에 집중하여 개발자의 학습과 코드 품질 향상을 지원해야 한다[2]. 그러나 LLM 은 제한된 컨텍스트 윈도우로 인해 집중력이 저하되고, 장기간 비연속적인 대화에서 중요한 정보가 대량의 무관한 정보 속에 묻혀 어텐션 메커니즘이 저하되는 문제가 있다[3]. 코드 리뷰 환경에서도 마찬가지로 기존 RAG 방식만으로는 누적되는 코드 리뷰의 연속성을 충분히 반영하지 못한다[4]. 본 연구에서는 이러한 문제를 해결하기 위해 RAG 와 메모리 주입 시스템을 결합한 하이브리드 아키텍처를 제안하고 실제 구현하여 그 효과성을 파일럿 적용을 통해 검증한다.

2. 본론



(그림 1) 시스템 아키텍처

2.1 시스템 아키텍처

본 연구에서 제안하는 AI 코드 리뷰 시스템은 두 가지 핵심 생성 메커니즘으로 구성된다. 첫째, (그림 1) Retrieval-Augmented Generation 의 연한 파란색 표

시 부분은 리뷰 품질 향상을 위한 최신 코드베이스 유지를 위해 증분 청킹 및 임베딩하여 코드를 주기적으로 벡터 데이터베이스에 저장하고 진한 파란색 표시 부분은 Retrieval Machine 을 통해 연관 코드 스니펫을 검색한다. 이때 연관 코드 스니펫 검색 함수는 OpenAI 의 Function Calling 도구로 구현되어 에이전트가 스스로 호출한다. 둘째, (그림 1)의 Memory-Injected Generation 부분은 장기간 코드 리뷰의 일관성 보장을 위해 이전 리뷰 내용을 요약하여 메모리에 저장하고 다음 리뷰에 활용한다.

2.2 워크플로우

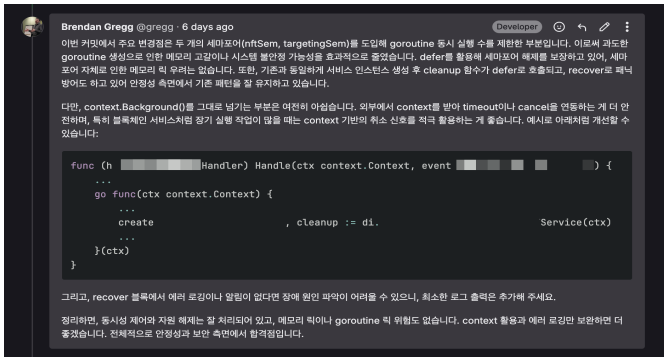
AI 코드 리뷰 시스템의 워크플로우는 다음과 같다. 클라이언트가 PR(Pull Request) 생성 및 커밋 푸시 시 GitLab 웹훅을 통해 HTTP 요청이 트리거되고, API 서버는 Git Hunk 를 파싱하여 변경 전후 코드와 diff 정보를 프롬프트에 포함한다. 첫 번째 PR 이 아닌 경우 Mem0 에서 이전 리뷰 기록을 로드하여 컨텍스트로 활용한다. 에이전트는 search_code 도구를 호출하여 연관 코드 스니펫을 검색하고 코드 리뷰를 수행한다. 생성된 리뷰는 JSON 형식의 점수와 리뷰 내용으로 구분되어 7 점 이상 시 PR 이 자동 승인된다.

2.3 프롬프팅 전략

비용최적화를 위해서 프롬프트 캐싱 전략을 도입하였다. OpenAI 는 프롬프트의 토큰이 1,024 개 이상일 경우 캐싱 기능은 자동 활성화되며, 정적 내용을 상단에 배치하여 캐시 적중률을 높였다. 이는 반복적인 API 호출 비용을 절감한다.

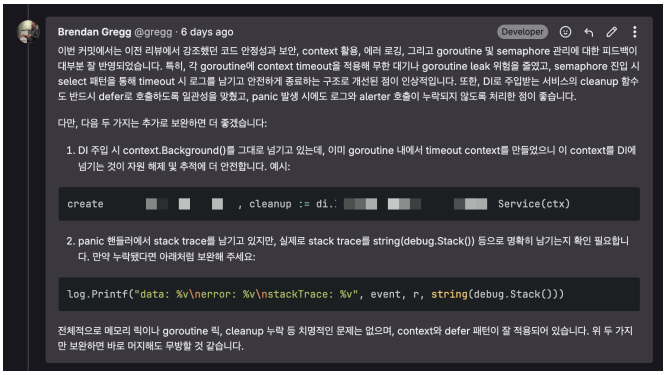
3. 파일럿 결과

코드 리뷰 품질과 일관성 유지 확인을 위해 PR 을 오픈하고 두 개의 커밋을 푸시하였다.



(그림 2) 첫 번째 코드 리뷰

첫 번째 커밋의 주요 변경사항은 고(Go)채널을 이용한 세마포어 구조체를 추가하여 동시 실행되는 고루틴의 수를 제한하는 동시성 제어 로직을 구현하였다. 이에 대해 에이전트는 (그림 2)와 같이 코드 변경사항을 정확히 분석하여 설명한다. 또한 컨텍스트 백그라운드를 파라미터에 직접 전달하는 대신 타임아웃이나 캔슬 기능을 연동할 것을 권고하는 실용적인 개선안을 제공하였다.



(그림 3) 두 번째 코드 리뷰

두 번째 커밋에서는 첫 번째 코드 리뷰를 반영하여 context 기반 타임아웃 처리를 적용하였다. 에이전트는 (그림 3)과 같이 이전 리뷰의 권고사항이 적절히 반영되었음을 긍정적으로 평가하였다. 그리고 불필요한 리소스 점유를 최소화하기 위해 defer 구문을 통한 context 취소 처리를 추가적인 개선사항으로 제안하였다. 이는 시스템이 단일 커밋 분석을 넘어 연속적인 변경사항에 대해 일관성 있는 리뷰를 수행할 수 있음을 보여준다.

4. 결론 및 향후계획

본 연구에서는 RAG 와 Memory-Injected Generation 을 결합한 하이브리드 AI 코드 리뷰 시스템을 제안하고 GitLab 환경에서 구현하였다. 제안된 시스템은 상용 LLM 을 활용하면서도 전체 코드베이스의 맥락을 효과적으로 제공하여 일관성 있는 코드 리뷰를 수행할 수 있음을 확인하였다. 현재 에이전트 기반 접근법은 컨텍스트 윈도우(30,000 토큰) 제한이 있으나, 향후 멀티-에이전트 아키텍처 확장 가능성을 고려하여 설계를 유지하였다. 또한 코드 이해를 위해서는 단순한 텍스트 유사성을 넘어 함수 호출 그래프와 같은 코드의 구조적 종속성 정보가 중요하다[5]. 이는 RAG 방식이 의미적 유사성 기반 검색에는 효과적이지만 코드의 구조적 종속성을 충분히 반영하지 못하는 한계와 관련이 있다. 향후 연구에서는 AST 기반 GraphDB 와 RAG 를 결합하여 코드 구조적 종속성을 반영하는 멀티 에이전트 아키텍처를 구현할 계획이다.

참고문헌

[1] 박희중 et al., “LLM 을 활용한 항공사 고객서비스 에이전트 시스템”, 한국정보기술학회 2024.11

[2] Zeeshan Rasheed et al., “AI-powered Code Review with LLMs: Early Results”, arXiv preprint arXiv:2404.18496

[3] Prateek Chhikara at al., “Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory”, arXiv preprint arXiv:2504.19413

[4] Toshihiro Kamiya, “A RAG Method for Source Code Inquiry Tailored to Long-Context LLMs”, arXiv preprint arXiv:2404.06082

[5] Asif Haider at al., “Prompting and Fine-tuning Large Language Models for Automated Code Review Comment Generation”, arXiv preprint arXiv:2411.10129