Trip Report for Attending FSE25

2025. 06. 23. ~ 06. 27.

최민석

FSE (Foundations of Software Engineering)는 소프트웨어 공학 전반을 다루는 대표적인 연례 국제 학술대회로, 프로그램 분석, 테스트, 버그 탐지 등 다양한 주제를 포함한다. FSE 2025는 노르웨이 트론헤임에서 개최되었고, 제출한 Demonstrations Paper가 붙어 발표할 수 있게 되어 참석했다.

10 Years Later - Revisiting How Developers Search for Code 소프트웨어 개발자들의 코드 검색 양상이 10년 후에 어떻게 변화했는지를 구글의 Code Search 도구를 중심으로 분석한 연구이다. 저자는 과거의 연구를 확장하여, 30개월간의 로그 데이터와 설문조사를 바탕으로 현재 개발자들이 코드 검색을 사용하는 이유를 조사했다. 특히 AI 도구 등 개발 환경의 변화가 코드 검색 사용에 미친 영향을 중점적으로 보았다. 코드 검색은 여전히 개발자들이 자주 사용하는 도구였으며, AI 도입에도 불구하고 사용 빈도는 줄지 않았다. 이는 AI 도구가 코드 검색을 대체하지 않고, 오히려 상호 보완적으로 활용될 수 있다는 점을 보여준다, 특히 대규모 프로젝트일수록 코드 검색은 코드 이해를 위한 중요한 방법이었다.
오늘날의 AI가 강력함에도 불구하고 실제 개발자들이 여전히 코드 검색을 적극적으로 활용한다는 점이 흥미로웠다. “혹시 AI가 가진 제한된 context window 때문에 (특히 대규모 코드 베이스에서) AI를 이용해 자동으로 작업하기가 어려워서 그랬을까?”, 하는 생각이 들었다. 그런 점에서 코드 검색과 정적 분석 및 프로그램 분석 도구를 결합하여 AI에게 더 정교한 컨텍스트를 제공하는 방향의 연구도 한 가지 방향이 될 수 있다고 생각했다. 예를 들어 AI에 요청하는 내용에 대해 코드 검색으로 클래스의 사용 예제를 찾고, 프로그램 분석 도구를 통해 그 주변의 호출 관계 등을 추출하여 자동으로 AI에 전달한다면, 코드 검색과 AI, 프로그램 분석의 융합을 통해 개발자 경험을 향상할 수 있을 것 같다.

An Empirical Study of Code Clones from Commercial AI Code Generators AI Code Generator에서 발생하는 코드 클론을 조사한 연구이다. 연구진은 GitHub의 오픈소스 저장소를 기반으로 84,000개의 프롬프트를 구축하고, 이를 통해 각 모델이 생성한 코드를 유형별 코드 클론 탐지 도구로 분석하였다. 그 결과, Java에서 약 7.5%, C와 Python에서 각각 약 3.3%와 4.7%의 Type-1 및 Type-2 코드 복제가 확인되었다. 일부 코드는 버그나 보안 취약점을 그대로 포함하거나 라이선스를 침해할 소지가 있었다. 내가 받아 들이기에는 라이언스 문제는 별로 와닿진 않았지만, 보안 취약점 쪽은 공감이 되었다. 요즘 바이브 코딩으로 내가 쓸 도구들을 만드는 일이 종종 있는데, 웹 쪽 코딩은 그래도 경험이 있어서 그런지 AI가 보안 취약점이 포함된 코드를 꽤 많이 생성하는 것이 눈에 보였다. AI가 작성한 코드라고 해서 보안적으로 더 안전할 이유가 (지금으로선) 별로 없어 보이기 때문에, AI가 앞으로 개발자의 역할을 상당 부분 대체하더라도 코드 클론 탐지와 같은 보안 관련 기술은 여전히 매우 중요한 과제로 남을 것 같다.

ChangeGuard: Validating Code Changes via Pairwise Learning-Guided Execution 이 논문은 소프트웨어 개발 과정에서 코드 변경이 의도한 대로 동작의 의미를 보존하는지를 자동으로 검증하기 위한 방법을 제안한다. 코드 리팩토링이나 성능 개선과 같이 의미를 변경하지 않아야 하는 수정이 실제로는 코드의 동작을 바꾸는 경우가 있지만, 기존의 회귀 테스트로는 알기 어렵다. ChangeGuard는 머신러닝 기반의 pairwise learning-guided execution 기법을 활용하여 코드 변경 전후의 실행을 나란히 수행하고, 주어진 입력에 따른 동작 차이를 분석함으로써 의미 변경 여부를 판단한다. 그 결과 ChangeGuard는 의미 변경을 77.1%의 precision과 69.5%의 recall로 검출하였으며, 기존 회귀 테스트의 recall(7.6%)을 크게 앞섰다. 또한 LLM이 생성한 코드 중 상당수가 의도치 않게 의미를 변경함을 확인함으로써 ChangeGuard의 실질적인 활용 가능성을 보였다.

나는 JavaScript Transpiler의 변환 규칙 검증을 연구하고 있어서 이 발표를 흥미롭게 보게 되었다. 비교 대상이 되는 프로그램이 동일한 언어로 작성되었고, 기본적으로 의미를 보존하는 것이 기대되는 변환이라는 점에서 내가 다루는 문제와 유사성이 컸다.
기존에는 JavaScript Transpiler의 규칙이 상대적으로 단순하기 때문에, 동적 실행 기반의 검증보다는 정적 분석으로 규칙 자체를 직접 증명하는 것이 유일한 방법이라고 생각했다. 그러나 ChangeGuard가 자동 리팩토링 도구를 대상으로 의미 변경 여부를 동적으로 검증하는 방식을 보고 나니, 만약 나도 JS Transpiler가 아니라 JS 리팩토링 도구를 타깃으로 한다면 결함을 fuzzing을 통해 탐지하는 연구 방향도 가능하지 않을까 하는 생각이 들었다. 예를 들어 Next.js는 버전을 올리면 deprecated된 기능을 codemod로 자동 변환하는 기능을 제공하는데, 이를 대상으로도 의미 변경 여부를 검증할 수 있을 것이다. (이 경우에는 웹 프레임워크의 내부 기능까지 돌아야 하기 때문에 ESMeta만으로는 한계가 있을 수 있지만) 다른 codemod 도구들까지 조사해보면 충분히 연구 가능성이 있는 영역으로 보인다.

The Struggles of LLMs in Cross-Lingual Code Clone Detection 다중 언어 간 코드 클론 탐지에 대해 다루는 연구다. 저자들은 LLM과 임베딩 모델 각각 문제를 얼마나 잘 해결할 수 있는지 비교했다. 특히, GPT-3.5-Turbo, LLaMA2 등 LLM과 8종의 프롬프트를 평가하고, GraphCodeBERT 등 임베딩 모델과 비교 실험을 수행했다. 실험 결과, LLM은 단순한 코드에서는 최대 0.99의 F1-score를 기록하며 우수한 성능을 보였으나, 복잡한 코드 문제에서는 성능이 크게 저하되었고 코드 클론의 개념을 일관되게 이해하지 못하는 한계가 드러났다. 반면, 텍스트 임베딩 기반 모델은 긴 코드가 많은 데이터셋에서 최대 20% 더 좋은 성능을 보였다.
연구는 LLM이 다중 언어 간 코드 클론 탐지를 기본적으로 잘 하긴 하지만, 기대만큼의 추론 능력을 발휘하지 못하고, 대신 임베딩 기반 표현 학습이 효과적임을 보여주었다. 이 결과가 NLP 외에도 전통적인(?) 요약 실행 기반 프로그램 분석 접근법이 여전히 중요한 역할을 할 가능성을 시사한다고 생각한다. 예를 들어, Debun이 JavaScript 프로그램에서 트랜스파일러가 건드리기 어려운 언어 기능을 바탕으로 불변식을 찾아 라이브러리 사용을 식별한 것처럼, 동적 언어들이 공통적으로 가지는 유사한 언어 기능을 이용해 이런 방식으로 다중 언어에 대한 코드 클론 감지 기법을 확장할 수 있을지 궁금하다. 각 언어마다 대응해주는데 드는 비용이 만만치 않기 때문에 쉽지 않겠지만, 해볼 만한 가치가 있는 것 같다.

발표 (JSSpecVis: A JavaScript Language Specification Visualization Tool) 후기 준비가 사실 출국할 때도 완전히 끝나지 않아 걱정이 많았지만, 학회 중간에도 자료를 보강하고 슬라이드를 수정하면서 꾸준히 연습한 덕분에 발표할 때는 버벅이지 않고 할 수 있었다.
발표 세션은 Program Analysis 4였는데, 학회장이 다른 세션과 떨어져 있어 청중이 적을까 걱정했지만, 한국에서 온 연구자분들도 와 주시고 예상보다 분위기가 좋았다. 질문도 많이 나오진 않았지만, 세션 진행자 분이 “실제로 자바스크립트 스펙을 프로그램처럼 놓고 실행하는 게 맞냐”는 질문을 주셨는데, 기계화 명세라는 개념이 아직 생소할 수 있어서 필요한 질문이었다고 생각했다.
이번 발표는 데모 트랙 페이퍼였던 만큼, 가능한 많은 사람들이 이 툴을 직접 써보길 바랐는데 실제로 발표 후 다운로드 수가 조금 늘어난 걸 보고 보람을 느꼈다. 앞으로도 더 많은 사람들이 각자의 목적대로 이 도구를 사용해볼 수 있길 바란다.