Journal of the Korean Institute of Industrial Engineers
Vol. 49, No. 3, pp. 209-222, June 2023.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2023.49.3.209

네트워크 확장을 통한 GCN 기반 GitHub 코드 리뷰어 추천 시스템

전병민[1] · 금영정[2†]

[1]서울과학기술대학교 데이터사이언스학과 / [2]서울과학기술대학교 산업공학과/데이터사이언스학과

GCN-based Reviewer Recommendation in Github based on the Network Expansion

Byeongmin Jeon[1] · Youngjung Geum[2]

[1]Department of Data Science, Seoul National University of Science and Technology (SeoulTech), South Korea
[2]Department of Industrial Engineering/ Department of Data Science, Seoul National University of Science and Technology (SeoulTech), South Korea

Github is an open-source platform which focuses on collaboration based on full-request. As an important task of the Github collaboration platform, a code review is critically important, as it plays a key role in the collaboration ecosystem as well as the software quality improvement. Therefore, there have been several works to recommend proper reviewers to improve the Github collaboration ecosystem. Despite the fact, however, previous studies have some common limitations. What is at the core in previous works is to focus on the recommendation of the existing network only, not considering the chances of extending the network in order to consider more proper candidates. For this purpose, this study suggests a GCN-based link prediction based on the network extension, which expands the boundaries of potential reviewer candidates in the network. As a result, the model based on the network extension shows a good performance compared to the existing network. In addition, we conducted a comparison for the existing reviewers and new reviewers, and identified the new and potential reviewers as having a positive impact compared to the existing reviewers.

Keywords: Github, Code Review, Recommendation System, Network Analysis, Link Prediction

1. 서 론

오픈 소스 소프트웨어(OSS)는 회사나 개인이 배포하고 수정할 수 있는 소프트웨어로 주로 컴퓨터 과학 분야 사용자 커뮤니티에서 인기가 있어 많은 개발자들로부터 큰 관심을 받고 있다(Chatziasimids & Stamelos, 2015). 깃허브(GitHub)는 이러한 오픈 소스 소프트웨어를 공유하는 가장 큰 규모의 플랫폼이다. 따라서 깃허브에서는 코드 및 프로젝트의 공유 및 수정 등 다양한 협업이 이루어지고 있고, 이에 대한 많은 연구가 진행되었다(Dabbish *et al.,* 2012; Zöller *et al.,* 2020). 깃허브에서의 협업은 풀-리퀘스트(Pull-Request)로 이루어진다는 것이 특징이다(Yu *et al.,* 2016). 풀-리퀘스트는 사용자가 프로젝트를 수정한 후 반영을 요청하는 과정으로 프로젝트의 소유자 외에도 모든 사용자가 협업에 참여할 수 있다는 특징이 있다. 사용자는 관심 레포지토리에 대해 손쉽게 접근하여 코드 수정 및 변경 사항 검토 등의 기여를 할 수 있다(Yu *et al.,* 2016). 풀-리퀘스트 과정에서 사용자를 반영을 요청한 요청자, 코드를 검토할 리뷰어로 구분할 수 있고, 이 중 리뷰어는 자발적으로 참여한 리뷰어와 요청을 받아 배정된 리뷰어로 나뉜다. 요청자는 리뷰어의 피드백을 통해 새롭게 수정한 사항을 다시 요청

본 연구는 한국연구재단 보호연구(과제번호: NRF-2020R1I1A2070429) 및 중견연구 (과제번호: 2023R1A2C1004752)의 지원을 받아 수행되었음.
본 논문은 제 1저자인 전병민 석사과정의 학위논문을 기반으로 작성되었음.
† 연락저자 : 금영정 부교수, 01811 서울과학기술대학교 산업공학과/데이터사이언스학과, Tel : 02-970-6528, Fax : 02-974-2849,
E-mail : yjgeum@seoultech.ac.kr
2023년 3월 8일 접수; 2023년 5월 10일 수정본 접수; 2023년 5월 12일 게재 확정.

할 수 있고, 이 과정에서 더 다양한 리뷰와 의견 및 토론을 촉발한다(Yu *et al.,* 2016). 소유자는 리뷰와 의견을 참고하여 최종적으로 요청을 반영할지를 결정한다. 이런 풀-리퀘스트는 협업을 위한 중요한 패러다임이다(Zhang *et al.,* 2022). 따라서 깃허브 내에서 풀-리퀘스트 기능은 활성화되어있고, 이런 풀-리퀘스트에 요청에 적절한 리뷰어를 추천할 수 있다면 검토를 조금 더 효율적으로 수행할 수 있을 것이다. 실제로 시간적인 측면과 깃허브 생태계 활성화의 측면에서 적절한 리뷰어 추천의 효과가 강조된 바 있다(Yu *et al.,* 2014). 또한 적절한 코드 리뷰가 품질향상 및 결함 발견에 도움을 준다는 연구도 존재했다(Cetin *et al.,* 2021). 이렇듯 다양한 방면에서 봤을 때, 적절한 리뷰어의 추천의 중요성은 강조되어왔다.

대부분의 깃허브 코드 리뷰어 추천 시스템은 관계 기반, 콘텐츠 유사성 기반으로 나뉜다. 관계 기반 추천 방식에서 대표적으로 Liao *et al.,*(2020)과 Ying *et al.,*(2016)의 연구는 리뷰어 간 연결관계를 바탕으로 추천하였다. 관계 기반 방식은 현재 네트워크 상에서 연결되어 있는 리뷰어들을 바탕으로 향후 연결될 가능성이 높은 리뷰어를 찾는다는 점에서 소셜 네트워크의 특성을 중점적으로 반영한 추천 방식이다. 반면 콘텐츠 유사성 기반 방식은 리뷰어의 특성, 요청자와 리뷰어 간 콘텐츠 유사성을 바탕으로 리뷰어를 추천해주는 방식으로 파일 명, 코드라인 변경 이력, 파일 경로, 제목의 유사성 등이 활용되었다. 즉 리뷰 대상이 되는 코드나 콘텐츠가 대상 리뷰어가 그 동안 했던 활동과 얼마나 유사한지를 파악하기 때문에 리뷰어의 전문성 및 활동에 기반하여 좀 더 정확한 추천을 할 수 있다는 장점이 있다.

이렇듯 선행 연구에서 관계 기반 및 콘텐츠 유사성 기반 접근을 사용하여 리뷰어 추천이 이루어져 왔으나, 기존 연구에는 크게 두 가지 한계점이 있다. 먼저 첫 번째 한계점은 리뷰어 추천에 활용되는 네트워크 범위가 기존 네트워크에 한정되어 있다는 점이다. 선행 연구에서 활용한 방식을 살펴보면 추천을 통해 선택되는 리뷰어가 레포지토리에 직접 관련된 유저로 한정되어 왔다(Yu *et al.,* 2015; Kononenko *et al.,* 2018). 하지만 이 경우 추천 가능한 리뷰어 수가 매우 적고, 리뷰어 풀이 한정된다는 한계가 있다. 하지만 레포지토리의 소유자가 아니지만 전문성이 높아 코드 리뷰가 가능한 사용자도 존재하기 때문에, 이런 사용자들을 추천해준다면 소프트웨어 품질 개선을 기대할 수 있으며, 나아가 깃허브 생태계에 도움이 될 수 있다.

두 번째 한계점은 관계 특성 및 콘텐츠 특성을 동시에 고려하는 접근이 부족했다는 점이다. 깃허브 내 협력이 일반적으로 잘 알려진 사용자 간 일어난다는 점에서 사용자 간 관계를 이용하는 것은 의의가 있으며, 동시에 콘텐츠 유사성에 대한 지표 또한 선행 연구에서 준수한 성능을 보여 활용 가치가 있다. 일부 연구에서 관계 특성과 콘텐츠 특성을 모두 변수로 활용한 경우가 있기는 하지만(Jiang *et al.,* 2017), 지표 수준이 아니라 방법론 차원에서 두 가지 측면을 동시에 반영할 수 있는 접근을 제시한다면 보다 체계적인 예측이 가능할 것으로 보인다.

본 연구는 기존 연구의 한계점들을 개선하기 위해 리뷰어 풀을 확장하기 위한 확장 네트워크를 구축하는 프레임워크를 제안하고 이를 통해 추천을 수행한다. 타겟 레포지토리를 선정하고, 해당 레포지토리의 풀-리퀘스트를 요청한 요청자와 리뷰어의 정보를 수집하여 기존 네트워크를 구성한다. 기존 네트워크의 리뷰어의 정보들을 활용하여 유사한 도메인의 레포지토리를 선정, 추가 네트워크를 구성한다. 최종적으로 기존 네트워크와 추가 네트워크를 결합하여 확장 네트워크를 구성한다. 결론적으로 확장 네트워크에서의 링크예측을 통해 기존 네트워크 속에서 등장하지 않았던 새로운 노드를 추천한다. 또한 관계 기반 방식과 콘텐츠 유사성 기반 방식을 동시에 활용하여 새로운 리뷰어를 추천하기 위해 두 가지 방식을 모두 사용할 수 있는 GCN 모델 기반의 링크예측을 선택하고 이를 통해 추천을 수행한다.

연구는 다음과 같이 구성된다. 제2장에서는 코드 리뷰어 추천 시스템과 관련된 선행 연구를 살펴본다. 제3장에서는 연구 방법과 프레임워크를 서술한다. 제4장에서는 제시한 연구 방법을 통해 사례 연구를 수행하고 결과를 분석한다. 제5장에서는 본 연구를 통해 얻을 수 있는 결론 및 의의, 그리고 한계점과 추후 연구 방안에 대해 서술한다.

2. 이론적 배경

2.1 코드 리뷰어 추천

코드 리뷰는 오픈소스 소프트웨어의 프로젝트에서 주로 이루어지며 소프트웨어의 품질을 보증한다 (Wessel *et al.,* 2020). 대규모 오픈소스 소프트웨어 플랫폼인 깃허브에서도 코드 리뷰는 활발하게 이루어지고 있다. 깃허브에서는 풀-리퀘스트 과정에서 요청자가 변경사항에 대해 반영을 요구할 때 이를 검토하기 위해 코드 리뷰가 이루어진다. 변경사항을 도입할 때 정상적으로 작동하는지, 불필요하거나 개선이 필요한 코드가 없는지, 최적화가 이루어지는지 등을 검토하기 위하여 코드 리뷰는 필수적이다. 따라서 이러한 중요한 절차인 코드 리뷰를 적절하게 수행할 수 있는 코드 리뷰어를 추천해주는 연구가 활발히 진행되었다.

기존 코드 리뷰어 추천 연구에서 활용된 방식은 콘텐츠 기반, 관계 기반으로 크게 분류될 수 있다. 콘텐츠 기반 방식은 요청자가 변경한 내용에 대해 리뷰어의 이전 리뷰 내용과의 유사성, 도메인 유사성 등을 활용하여 리뷰어를 추천해주는 방식이다. Jeong *et al.,*(2009)는 요청자 정보, 파일명, 파일 및 수정 코드라인 수, 코드 수정 내용 등 지표를 통해 요청이 반영 여부를 분류하는 모델을 제시함과 동시에 상위 5명의 적합한 리뷰어를 예측했다. Balachandran(2013)은 코드라인의 변경 이력을 바탕으로 코드의 검토와 요청의 승인, 반려 과정을 자동화한 ReviewBot과 파일 변경 이력을 사용한 RevHistRECO를 제안했다. Thongtanunam *et al.,*(2015)은 요청 파일의 저장경로와 리뷰어의 이전 리뷰 파일의 저장경로간 유사성을 측정하여

리뷰어를 추천했다. 이는 요청자의 요청 분야와 리뷰어의 전문 분야 간의 유사성을 활용했다고 볼 수 있고 79%의 준수한 정확도를 보였다. Jiang *et al.,*(2015)은 요청자와 리뷰어 간 library 및 technology의 유사성을 활용했다. Yu *et al.,*(2016)은 검토 요청의 제목 및 내용과 리뷰어가 이전에 리뷰한 리뷰의 제목과 내용의 유사성을 TF-IDF로 계산하여 리뷰어를 추천했다.

관계 기반 추천 방식은 요청자와 리뷰어 간 관계, 즉 이전에 얼마나 교류 및 상호작용을 했는지를 바탕으로 추천하는 방식이다. Jiang *et al.,*(2015)은 리뷰어와 요청자 간 following 관계, 해당 요청자의 풀-리퀘스트에 대해 해당 리뷰어가 몇 번이나 처리했는지 등의 지표를 사용했다. Yu *et al.,*(2016)은 요청자와 리뷰어가 mention, 댓글 등을 통해 서로 언급한 정도를 활용하여 연관된 정도를 활용했다. 특히 이를 보다 확장하여 추천할 리뷰어의 활동력을 바탕으로 요청자와 리뷰어의 관계를 정의하고 이를 바탕으로 추천하는 방식도 활용되어 왔다. Jiang *et al.,*(2015)은 리뷰어가 리뷰한 풀-리퀘스트 수, 리뷰에 걸리는 시간 등을 추가 지표로 활용했다. 이러한 관계기반의 방식은 활발한 교류와 협업이 이루어지는 깃허브의 특성을 잘 활용할 수 있고, 본 연구에서 활용할 가치가 있다.

콘텐츠 기반 및 관계 기반 방법을 결합하여 활용한 연구도 존재했다. Jiang *et al.,*(2017)은 텍스트 유사성, 리뷰어가 이전에도 요청자의 요청을 검토했는지 여부, 리뷰어의 댓글 빈도를 지표로 사용해 콘텐츠 기반 및 관계 기반 방식을 모두 지표로 활용한 바 있다. 그러나 지표 수준이 아니라 방법론 차원에서 이들 두 가지 접근을 동시에 반영할 수 있다면 보다 체계적인 예측이 가능하다. 본 연구에서는 콘텐츠 기반 방식과 관계 기반 방식을 보다 체계적으로 활용하기 위해 GCN 모델을 사용한다. GCN의 노드 정보를 통해 콘텐츠 유사성을 사용하고, 노드 간 연결 관계를 통해 관계성을 활용할 수 있기 때문에 두 가지 관점을 모두 고려할 수 있다.

코드 리뷰어 추천의 두 번째 한계점은 추천해주는 리뷰어가 레포지토리의 소유자 및 리뷰어의 지인 등으로 한정되어 있다는 점이다. 즉 추천할 수 있는 리뷰어의 풀은 정해져 있고, 추천의 대상이 될 리뷰어의 수 자체가 제한적이었다. 깃허브의 특성상 어떤 사용자든 풀-리퀘스트를 요청 및 검토할 수 있지만 최종적으로 해당 요청을 프로젝트에 반영할지 여부를 판단하는 것은 레포지토리의 소유자이다. 따라서 소유자 일부만이 리뷰를 모두 담당하는 경우가 대부분이었고, 많은 요청에 대해 응답하는 시간이 지연되었다. 이러한 문제점을 개선하고자 응답 지연 요소를 찾고 풀-리퀘스트의 응답 시간을 단축시키는 연구가 선행됐다(Yu *et al.,* 2015; Kononenko *et al.,* 2018). 결론적으로 기존 연구의 목적은 요청자의 풀-리퀘스트 요청에 대해 검토 및 반영 여부를 빠르게 판단해줄 리뷰어인 소유자를 추천해주는 것이었다.

하지만 시간적인 측면 외에 프로젝트의 품질적 측면으로 보았을 때, 해당 방식은 리뷰어가 매우 한정적이기에 비효율적이다. 깃허브의 플랫폼적인 특성상 어떤 사용자든 풀-리퀘스트에 대해 검토하고 의견을 낼 수 있고, 따라서 소유자가 아니더라도 양질의 리뷰어들이 코드의 리뷰 및 코멘트를 통해 요청자의 코드를 개선할 수 있고 최종적으로는 레포지토리의 품질을 향상시킬 수 있다. 하지만 이런 새로운 리뷰어를 추천하는 방향의 연구는 자세히 고려되지 않았다. 따라서 본 연구에서는 네트워크 확장을 통해 기존에 존재하지 않는 새로운 리뷰어 풀을 추가하여, 이전에는 추천될 수 없는 완전히 새로운 리뷰어를 추천해주는 것을 목적으로 한다.

2.2 링크 예측

링크 예측은 그래프 구조에서 사용되는 방법론으로 그래프의 구조적, 특징 정보를 바탕으로 새롭게 연결될 링크를 예측하는 방법론(Lu *et al.,* 2011)으로 추천 시스템(Su *et al.,* 2020), 소셜 네트워크 관계 예측(Cannistraci *et al.,* 2013) 등의 다양한 분야에서 활용되었다. 이러한 링크 예측은 유사도 기반, 확률 기반, 학습 기반 방식으로 크게 세 가지로 분류할 수 있다.

유사도 기반 방식은 그래프의 구조를 활용하는 방식으로, 두 노드 간 유사성을 계산하여 임계값 이상일 때 연결된다고 예측하는 방식이다. 유사성 계산 지표로는 Jaccard Index, Adamic-Adar Index, Resouce Allocation Index, Cosin Similarity 등이 있다. 유사도 기반 방식은 비교적 간단한 방식으로 노드 간 연결을 예측할 수 있다는 장점이 있다.

확률 기반 방식은 네트워크의 분포에서 조건부 확률을 기반으로 링크의 연결 가능성을 예측한다. 이 방식은 복잡한 연산과정으로 인해 네트워크의 규모가 커질수록 연산량이 많아지고 비용과 시간이 많이 소모된다는 단점이 있다(AI Hasan and Zaki, 2011).

학습 기반 방식은 머신러닝 모델을 사용하는 방식으로 주로 의사결정나무, 나이브 베이즈, SVM 등의 지도학습 기반 모델을 활용한 이진 분류 모델로 링크의 연결 여부를 예측한다. 학습 기반 방식은 일반적으로 높은 성능을 보인다는 장점이 있다.(Kumar *et al.,* 2020). 최근에는 학습 기반 방식에서 그래프 데이터에 적합하게끔 딥러닝 모델을 적용한 그래프 인공 신경망(GNN, Graph Neural Network)이 등장했다(Ai *et al.,* 2022). 이러한 GNN은 그래프의 구조와 동시에 노드와 링크의 특징 정보를 모두 활용한다는 특징이 있다. 네트워크를 벡터로 임베딩한 후 임베딩된 네트워크에서 그래프 구조, 노드 간 유사도를 계산하여 그래프 분류, 노드 분류 및 링크 예측과 같은 다양한 분야에서 활용할 수 있다(Islam *et al.,* 2020). 그 중 GCN(Graph Convolution Network)은 가장 인기있는 모델로, 스펙트럴 그래프 이론을 기반으로 그래프에 CNN(Convolutional Neural Network)의 Convolution 연산 개념을 적용한 모델이다. 그래프 구조에서 특정 레이어만큼 필터를 이동하여 이웃 노드 정보를 학습하고, 이를 바탕으로 타겟 노드 정보를 업데이트한다. 이러한 GCN은 콘텐츠 기반 및 관계성 기반 지표를 모두 사용할 수 있고, 이를 통한 링크 예측을 통해 추천 시스템에 사용할 수 있어 본 연구에 적합한 모델이다. 따라서 본 연구에서는 GCN 모델을 활용한 링크예측을 통해 리뷰어 추천시스템을 제안한다.

3. 연구 방법

<Figure 1>은 본 연구의 프레임워크를 나타낸다. 크게 네트워크 확장, 링크 예측을 통한 추천, 예측 결과 비교 분석의 세 단계로 분류된다. 네트워크 확장은 새로운 리뷰어 풀을 확보하기 위한 과정으로 분석 대상이 될 기존 레포지토리의 네트워크를 구축한 후 기존 레포지토리와 유사한 도메인의 추가 레포지토리 네트워크를 구축하여 결합하는 방식으로 이루어진다. 이후 확장된 네트워크의 노드 정보와 연결 정보를 활용하여 GCN 모델을 통한 링크예측으로 리뷰어를 추천한다. 수집된 요청자와 리뷰어 데이터를 활용하여 노드 정보 행렬을 구성하고, 요청자와 리뷰어 간 연결 관계를 동시 발생 행렬을 이용하여 인접행렬을 생성한다. 생성된 노드 정보 행렬과 인접행렬을 GCN 모델에 입력하여 학습한 후 평가 데이터에 대해 링크 예측을 수행한다. 마지막으로 테스트 데이터에서 링크 예측을 통해 새롭게 생성된 링크, 즉 추천 결과를 각 특징에 맞게끔 기존 리뷰어와 새로운 리뷰어로 분류하여 그룹화하고, 그룹 간의 유의미한 차이가 있는지를 통계 분석을 통해 확인한다.

3.1 네트워크 확장

(1) 기존 레포지토리의 네트워크 구성

기존 네트워크는 분석 대상이 될 레포지토리의 정보로 구성되며 본 연구에서는 Rails/Rails 레포지토리를 사용하였다. Rails 레포지토리는 프로그래밍 언어 Ruby를 기반으로 만들어진 웹 애플리케이션 프레임워크에 대한 대규모 프로젝트로, 수많은 사용자의 협업이 이루어졌고, 많은 수의 풀-리퀘스트, 워치(watch), 포크(fork), 스타(star) 등의 데이터를 가지고 있기에 코드 리뷰어 추천에 적합하다. 따라서 기존에 코드 리뷰어를 추천하는 연구에서 주로 활용된 바 있다(Yu *et al.,* 2016; Yang *et al.,* 2018).

수집의 대상이 되는 데이터는 레포지토리의 풀-리퀘스트 정보, 요청자의 정보, 리뷰어의 정보이다. 깃허브 API를 통해 총 28,600개의 풀-리퀘스트 정보, 해당 풀-리퀘스트를 요청한 요청자와 할당된 리뷰어의 정보를 수집했다. 그 결과 중복되는 경우를 제거했을 때 요청자는 3,724명, 리뷰어는 719명, 참여자는 6,473명이었다. 코드 리뷰에 있어 리뷰어가 한정적인 것을 감안해도 요청에 비해 리뷰어의 수가 현저히 적은 것을 확인할 수 있다.

<Figure 2>는 실제 풀-리퀘스트의 예시를 보여준다. 이 경우 요청에 대해 리뷰어가 배정되지는 않았지만, 여러 사용자가 요청된 코드를 검토하고 토론을 거친 후 결과적으로 해당 요청이 반려되었음을 확인할 수 있다. 이 사례에서는 리뷰어가 배정이 되지 않아 코드리뷰가 이루어지지 않은 것이 아닌 해당 풀-리퀘스트에 의견을 남긴 모든 참여자(participant)들이 실질적으로 코드를 검토했다고 볼 수 있으며, 실제로 코드 검토 프로세스에 있어 토론에 참여한 사용자의 중요성을 강조한 바 있다(Yu *et al.,* 2016). 따라서 본 연구에서는 리뷰어의 개념을 풀-리퀘스트에서 배정된 리뷰어 뿐 아니라, 코드 리뷰 과정에 참여한 참여자로 확장하여 사용한다. 이후부터 용어의 의미 구분을 위해 기존 의미의 배정된 리뷰어를 배정 리뷰어로, 참여자를 리뷰어로 사용한다.

그 결과 리뷰어는 총 6,473명으로 배정된 리뷰어만 사용했을 때보다 크게 증가했으며, 요청자와 리뷰어간 링크는 총 28,565개가 존재했다. 이 중 요청자이자 리뷰어인 경우는

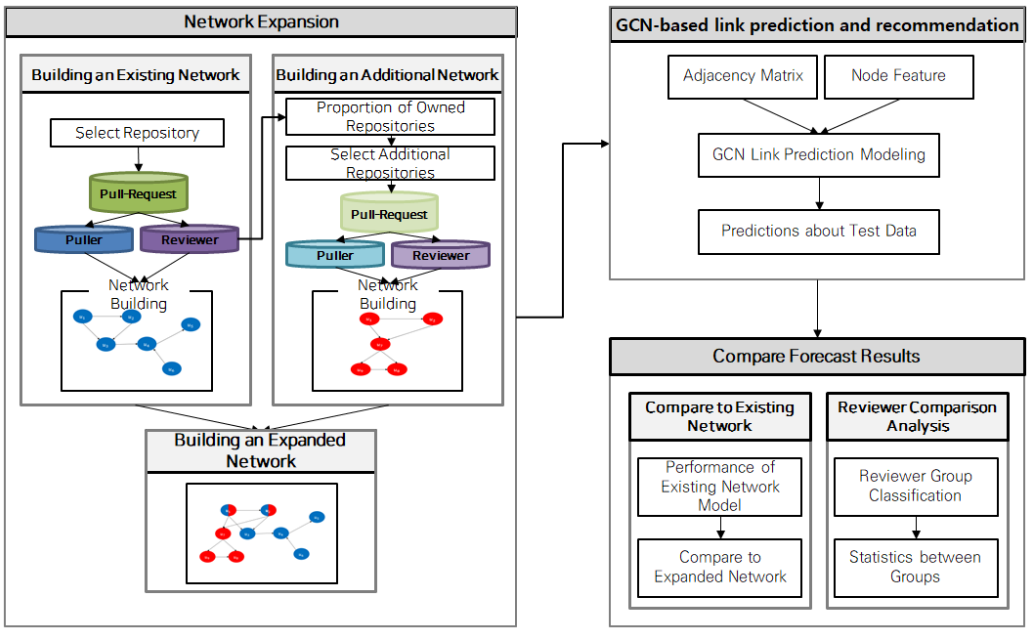

Figure 1. Framework of the Research

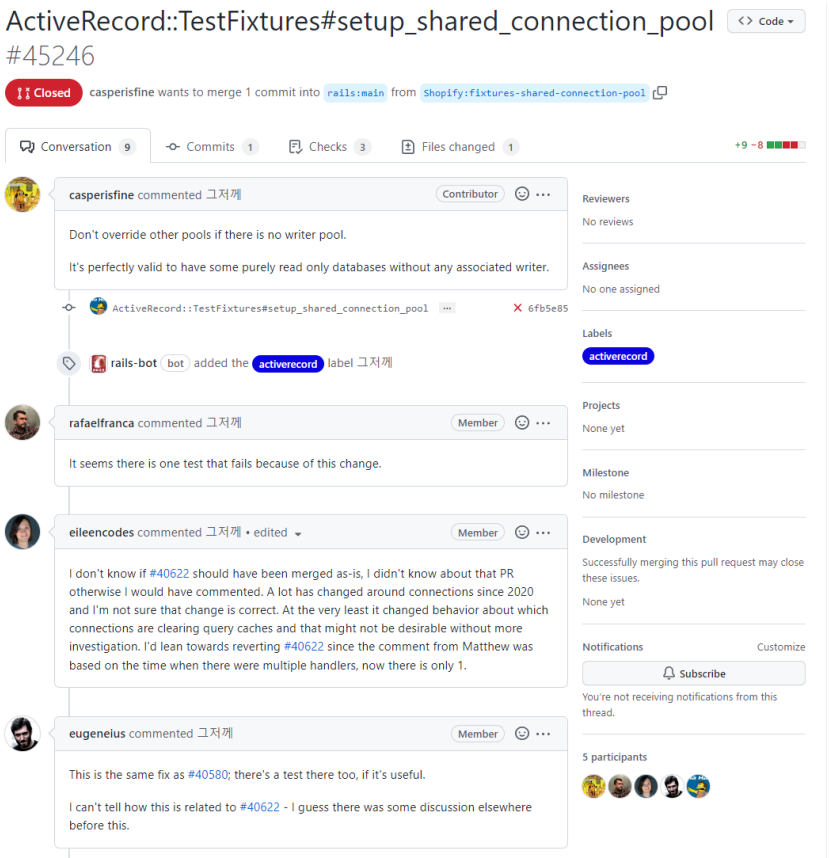

Figure 2. Example of a Pull-Request(From Github webside- https://github.com/rails/rails/pull/45246)

1,966명으로, 결과적으로 초기 네트워크에는 총 8,231명의 사용자가 존재한다. 다음 단계로 8,231명의 사용자의 정보를 수집한다. 깃허브 API와 Selenium을 활용한 웹 크롤링을 활용하여 사용자의 레포지토리 수, follower 수, following 수, 기준일로부터 1년간 commit 수, 소유한 레포지토리 등의 정보를 수집한다.

구성된 기존 네트워크 G_{org}은 수식 (1)과 같다. 사용자 노드 V에는 요청자와 리뷰어가 공존하고, 링크 E는 요청자의 요청에 대해 리뷰어가 리뷰했을 때 연결관계가 있는 것으로 정의한다.

$$G_{org} = (V_{org}, E_{org}) \quad (1)$$
$$V_{org} = \{u_1, u_{2\ldots}, u_n\}$$
$$E_{org} = \{(u_1,u_2),\ldots(u_1,u_n)\}$$

(2) 추가 레포지토리의 네트워크 구성

확장 네트워크를 구성하기 위해 우선적으로 추가 레포지토리의 네트워크를 구성한다. 추가 리뷰어는 기존 레포지토리의 리뷰 요청에 대해 답변할 수 있는 도메인 전문성이 필요하다. 따라서 해당 조건을 충족하기 위해 추가 대상이 될 레포지토리는 기존 레포지토리와 유사한 도메인의 레포지토리로 선정하는 과정이 필요하다. 이를 충족하기 위해 앞서 구축한 기존 네트워크의 기존 리뷰어들이 소유한 레포지토리를 활용한다. 소유한 레포지토리는 본인이 직접 등록하거나, 포크를 통해 다른 사용자의 레포지토리를 본인의 저장소로 가져온 경우 두 가지가 존재하며 두 경우 모두 사용자의 관심분야를 나타낸다. 따라서 소유한 레포지토리 정보를 통해 리뷰어의 관심 분야를 파악할 수 있다.

추가 네트워크 구성 과정의 예시는 다음과 같다. 기존 레포지토리의 리뷰어가 n명 있을 때, 리뷰어 집합을 $R = \{reviewer_1, reviewer_2, \ldots, reviewer_n\}$ 로 표현할 수 있고, 전체 레포지토리의 집합을 $Repository = \{repo_a, repo_b, \ldots, repo_m\}$로 표현할 수 있다. 이 때, 리뷰어1이 레포지토리 a,b를 소유한 경우 $reviewer_1 = \{repo_a, repo_b\}$로 표현이 가능하다. 해당 과정을 n명의 리뷰어에 대해 반복하여 리뷰어 별 소유하고 있는 레포지토리를 파악한다. 결과적으로 리뷰어가 몇 개의 레포지토리를 소유하고 있는지 파악할 수 있고, 반대로 특정 레포지토리를 몇 명의 리뷰어가 소유하고 있는지 파악할 수 있다. 즉, 각 레포지토리 별로 전체 리뷰어에 대해 연산을 수행한다. 계산 과정은 수식 (2)와 같이 지시함수(Indicator function)를 사용하여 표현한다. 특정 레포지토리가 리뷰어에게 소유되어 있으면 1, 소유되어 있지 않다면 0의 결과를 얻고, 해당 과정을 모든 리뷰어에 대해 반복하여 합산한다. 이후 해당 값을 수식 (3)과 같이 전체 리뷰어의 수로 나누어 특정 레포지토리를 리뷰어들이 소유한 비율 D_{repo}를 얻는

다. 결과적으로 D_{repo}가 높은 레포지토리일수록 기존 리뷰어들이 공통적으로 소유한 레포지토리로, 리뷰어들의 공통된 관심분야, 즉 기존 레포지토리와 유사한 도메인의 레포지토리임을 나타낸다.

$$C_{repo_k} = \sum_{reviewer \in R} I(repo_k \in reviewer) \quad (2)$$
$$repo_k \in Repository$$

$$D_{repo_k} = \frac{1}{|R|} C_{repo_k} \quad (3)$$

총 k개의 레포지토리에 대해 계산된 D_{repo}값에 따라 임계값을 지정하여 임계값 이상이 충족된다면 추가 레포지토리의 대상으로 선정한다. 이후 추가 레포지토리의 정보를 통한 추가 네트워크 구축 과정은 기존 네트워크의 구축 과정과 동일하게 이루어진다.

요청자와 리뷰어의 연결 관계를 통해 구성된 추가 네트워크를 수식으로 표현하면 (4)와 같다.

사용자 노드와 링크로 구성되는 구조는 기존 네트워크의 구조와 동일하지만, 사용자 노드 V를 구성하는 요청자와 리뷰어가 초기 네트워크의 요청자와 리뷰어에 더해 확장되어 추가된 경우가 있다는 점에서 차이가 있다.

$$G_{add} = (V_{add}, E_{add}) \quad (4)$$
$$V_{add} = \{u_1, u_{2\ldots}, u_m\}$$
$$E_{add} = \{(u_1,u_2),\ldots(u_1,u_m)\}$$

(3) 확장 네트워크 구성

최종적으로 분석에 활용될 확장 네트워크의 구성은 기존 네트워크와 추가 네트워크를 결합하는 방식을 통해 구성된다. 네트워크의 구조는 기존, 추가 네트워크의 노드, 링크 정보를 활용한다. 확장 네트워크를 표현한 수식은 (5)와 같다.

$$G_{expand} = ((V_{org} + V_{add}), (E_{org} + E_{add})) \quad (5)$$

<Figure 3>은 지금까지의 네트워크 확장 과정을 그림으로 설명한다. 결과적으로 확장된 네트워크는 두 개의 네트워크를 결합한 형태로, 기존 네트워크보다 더욱 다양한 노드, 링크를 포함한다. 또한 위 그림에서 볼 수 있듯 u_1, u_4, r_1 와 같이 네트워크 확장 이전의 기존, 추가 두 개의 네트워크에서 동시에 등장한 노드가 존재할 것이고, 해당 노드들의 연결 관계를 통해 확장된 네트워크에서도 구조적 특징을 활용한 링크 예측을 기대할 수 있다.

3.2 GCN 기반 링크예측 및 추천

앞서 구성된 네트워크를 통해 링크예측을 수행하여 새로운 리뷰어를 추천한다. 링크예측에는 GCN 모델이 사용된다. GCN은 CNN의 Convolution 연산 개념을 그래프 데이터 구조에 적용한 모델로 알고리즘의 수식은 (6)과 같다. 노드의 정보를 담은 행렬인 $H^{(l)}$를 입력하게 되면 은닉층의 수 l 만큼 행렬곱 연산과 활성화 함수를 거치는 과정을 반복하고 결과적으로 $H^{(l+1)}$를 얻게 된다. 이 과정에서 은닉층의 수에 따라 학습하게 될 이웃 노드의 범위가 정해진다. 예를 들어 은닉층의 수가 2이면 타겟 노드와 2홉(hop) 거리의 노드 정보까지 학습한다.

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \quad (6)$$

(1) 인접행렬 구성

인접행렬은 3.1에서 구성된 확장 네트워크의 구조적 정보를 통해 생성된다. 인접행렬은 (노드 수 × 노드 수)의 형태로 구

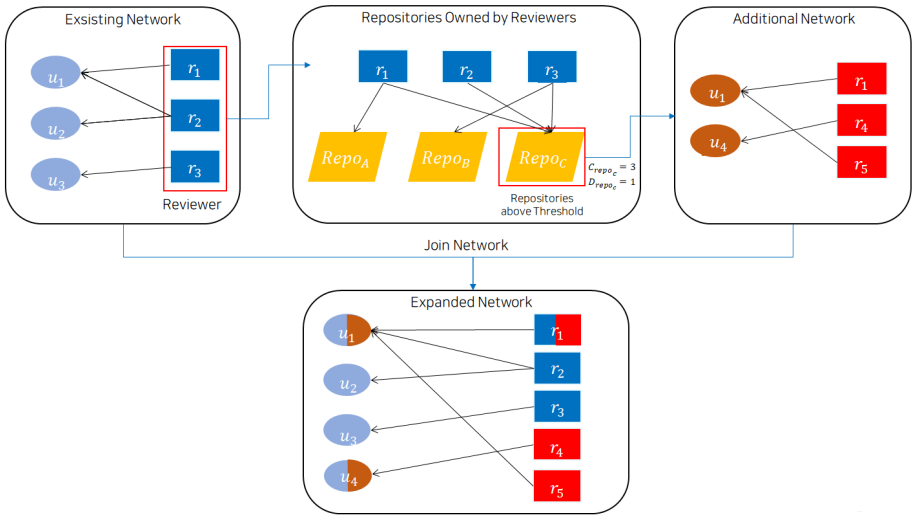

Figure 3. Network Extension Process

성된 정방 행렬로서 노드 간의 연결 정보를 나타낸다. 이때, 노드 간의 연결 여부만을 활용하여 1 또는 0으로 행렬을 구성하는 방법은 Unweighted 방식이고, 연결 수에 가중치를 부여하여 행렬을 구성하는 방법은 Weighted 방식이다. 본 연구에서는 기존 노드 간 연결이 몇 번 이루어졌는지에 대한 가중치가 예측에 중요하다고 판단하여 인접행렬을 Weighted 방식으로 구성한다.

(2) 노드 정보 변수 구성

본 연구는 노드 간의 연결관계 뿐 아니라, 노드 정보변수를 활용하여 콘텐츠 기반 방법을 모두 고려하였다. 고려된 노드 정보 변수를 정리한 표는 <Table 1>과 같다. 노드 정보 변수는 Repository, Follower, Following, Commit, Career, Main language의 총 6개로 구성된다. Repository는 사용자가 등록하거나 포크한 레포지토리 수를 의마한다. Follower와 Following은 각각 사용자의 Follower 수와 Following 수를 뜻하며, Commit은 사용자가 지난 1년 동안 저장소에 기여한 커밋 수를 말한다. Career는 깃허브를 사용한 경험을 나타내는 지표로 가입일로부터 10년 이상이면 Senior 사용자로, 미만이면 Junior 사용자로 분류한 후 원-핫 인코딩(One-Hot Encoding)을 통해 2차원 벡터로 활용된다. 앞선 5가지 지표는 사용자 활동의 지표라고 할 수 있다. Main language는 사용자의 주 프로그래밍 언어를 의미한다. 깃허브 프로필에서 사용자가 직접 자신의 프로그래밍 언어를 지정하는 경우는 매우 드물기 때문에 사용자의 주요 프로그래밍 언어를 지정하는데는 다른 방법이 필요하다. 따라서 본 연구에서는 사용자의 레포지토리 정보를 활용하였다. 사용자가 소유한 각 레포지토리에 사용되는 프로그래밍 언어를 나열한 후, 가장 많이 등장한 프로그래밍 언어가 사용자의 주요 프로그래밍 언어로 지정되었다. 그 결과 58개의 프로그래밍 언어가 등장하였으나, 이는 불필요하게 많은 수였다. 이에 프로그래밍 언어의 분포를 확인해본 결과 특정 언어들이 대부분을 차지했고, 이외의 언어들이 차지하는 비중은 매우 적었다. 따라서 분포의 1% 이상을 차지하는 7개 언어만을 사용하였고, 나머지 언어는 Others로 변환하여 총 8개의 범주로 나누어 원-핫 인코딩을 통해 벡터로 사용하였다.

(3) GCN 링크 예측 모델링

다음으로 인접행렬과 노드 변수의 목록을 GCN 모델에 적용한다. 우선 링크 예측을 위한 모델의 학습, 검증, 평가에 필요한 데이터를 분리할 필요가 있다. 우선 연결된 Positive link를 5:3:2의 비율로 나눈다. 일반적인 그래프 구조의 특성상 Negative link 수가 Positive link 수보다 압도적으로 많기 때문에 Negative link는 Negative sampling을 통해 Positive link 수와 동일하도록 샘플링한다. 이에 따라 전체 링크에 대한 학습, 검증, 평가 데이터의 비율은 5:3:2로 유지되며, Positive 링크와 Negative 링크의 비율도 동일하게 1:1로 활용된다.

위 과정을 거쳐 구성된 데이터를 통해 GCN 모델을 구축한다. 먼저, 모델에 구성된 숨겨진 레이어의 수는 Kipf and Welling(2016)의 연구에서 2~3개 일 때 가장 성능이 우수하다고 검증된 바 있다. 따라서 본 연구에서는 레이어의 수를 2개로 설정한다. 결과적으로 타겟 노드에서 2홉만큼 떨어진 노드의 정보를 학습하게된다. 은닉층을 통과할 때 활성화 함수는 ReLU를 이용하고, 출력 레이어에서는 Sigmoid 활성화 함수를 사용하여 링크 연결의 이진 분류를 수행한다. 이 과정을 수식으로 정리하면 (7)과 같다. 노드의 변수행렬이 GCN모델을 거쳐 각 노드에 대한 벡터 값으로 표현된다. 추후 링크 예측은 노드 쌍의 벡터 간 유사도를 통해 이루어진다.

$$Z = f(X,A) = sigmoid(\hat{A}ReLU(\hat{A}XW^{(0)})W^{(1)}) \quad (7)$$
$$\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$$

(4) 모델 평가

모델의 성능은 Figure 4와 같이 사전에 분리된 평가 데이터에 대해 링크의 연결 여부를 예측하는 이진 분류 모델로 평가한다. 예측의 정확도를 평가하기 위하여 정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F1 score를 활용했고, 각 수식은 (8), (9), (10), (11)과 같다. 정확도는 전체 데이터 중 정확하게 분류된 데이터의 비율을 의미한다. 정밀도는 분류 모델이 Positive로 예측한 경우 중 실제로 Positive인 비율을 뜻하고, 재현율은 실제로 Positive인 경우 중 분류 모델이 Positive로 예측

Table 1. Node Information Variable List

Vairable	Description	Type
Repository	Number of repositories	Int
Follower	Number of followers	Int
Following	Number of followings	Int
Commit	Number of commits per year	Int
Career	Separated into Senior and Junior based on the user's subscription date	Categorical(2)
Main language	Among the programming languages of the repository you registered with the programming language with the largest percentage	Categorical(8)

	Predicted Positive	Predicted Negative
Actual Positive	TP(True Positive)	FN(False Negative)
Actual Negative	FP(False Positive)	TN(True Negative)

Figure 4. Confusion Matrix

한 비율을 뜻한다. 정밀도와 재현율은 상호보완적이며 성능을 평가할 때, 동시에 고려되는 것이 바람직하다. F1 score는 정밀도와 재현율을 모두 활용한 것으로, 두 지표의 조화평균을 나타낸다.

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \tag{8}$$

$$Precision = \frac{TP}{TP+FP} \tag{9}$$

$$Recall = \frac{TP}{TP+FN} \tag{10}$$

$$F1\,score = 2 \times \frac{\mathrm{Precision} \times Recall}{\mathrm{Precision} + Recall} \tag{11}$$

3.3 예측 결과 비교 분석

마지막으로, 학습된 GCN 모델을 사용하여 평가 데이터를 예측한 후, 생성된 새로운 링크에 대한 분석을 수행한다. 새 링크가 실제로 요청자와 리뷰어 간의 링크인지 확인한 후, 요청자와 리뷰어를 기존 네트워크 또는 추가 네트워크 중 어느 네트워크에 속해 있던 노드인지를 식별한다. 그런 다음 리뷰어를 기존 네트워크에 속해 있던 기존 리뷰어 그룹과 추가 네트워크에 속해 있던 추가 리뷰어, 즉 잠재적 리뷰어 그룹으로 분류한다. 두 그룹을 비교하여 기존 리뷰어와 잠재적 리뷰어가 실제로 어떤 차이가 있는지 분석한다.

4. 연구 결과

4.1 기존 네트워크 구성

본 연구에서는 많은 풀-리퀘스트가 있는 레포지토리로 기존 코드 리뷰어 추천 연구에 사용되어온 Rails/Rails 레포지토리를 분석하였다. 수집할 데이터는 Rails 저장소의 풀-리퀘스트 정보이며, 총 28,600개의 풀-리퀘스트에 대해 GitHub API를 사용하여 요청자와 리뷰어 정보를 수집했다. 그 결과, 중복제거 시 수집된 요청자는 3,724건, 배정된 리뷰어는 618명, 리뷰어는 6,473명으로 총 8,231명의 사용자가 있었다. 이 사용자들의 레포지토리 수, 팔로워 및 팔로잉 수, 연간 커밋 수, 생성 날짜 및 소유 레포지토리 관련 정보(이름, 프로그래밍 언어 사용)를 수집했다.

기존 네트워크는 수집된 사용자의 연결을 활용하여 구성된다. 요청자의 풀-리퀘스트 요청을 검토한 리뷰어는 연결되었다고 판단하고 동시 발생 행렬을 통해 인접행렬을 생성한다. 일반적인 네트워크 기반 추천 시스템 연구에서는 항목과 사용자 간의 2-mode 네트워크, 즉 이분 그래프(Bipartite Graph) 형태의 네트워크 구조를 활용하여 사용자에게 항목을 추천한다. 그러나 본 연구의 경우 검토자와 요청자는 서로 변환할 수 있다. 예를 들어, 어떠한 코드를 리뷰한 리뷰어는 새로운 주제로 풀-리퀘스트를 요청할 수 있으며, 요청자 또한 다른 풀-리퀘스트를 검토할 수도 있다. 따라서 본 연구에서는 네트워크를 요청자와 리뷰어 간의 이진 그래프로 사용하는 것이 아니라 사용자 간의 1-mode 네트워크로 사용한다. 또한 리뷰어가 요청자의 요청에 대해 검토했음을 명확히 하기 위해 링크를 방향성이 있는 Directed graph로 사용한다. 이 때, 소스 노드는 코드를 리뷰한 사용자, 타겟 노드는 리뷰를 요청한 사용자가 된다. 사용자 간 연결된 횟수에 대한 가중치는 추천에 중요한 요소이므로, 링크에 가중치가 있는 Weighted network로 구성한다. 구성된 기존 네트워크는 중복을 제거한 후 노드 정보를 얻을 수 없는 7개의 노드를 제거하여 8,224개의 노드와 28,565개의 링크로 구성된다.

4.2 추가 네트워크 구성

(1) 추가 대상 레포지토리 선정

추가 네트워크를 구성하게 될 추가 대상 레포지토리는 기존 네트워크에서의 배정 리뷰어 정보를 통해 선정한다. 수집된 각 배정 리뷰어들이 소유한 레포지토리를 정렬한 후, 레포지토리별 D_{repo}값을 파악한 후 D_{repo} 값을 계산한다. <Table 2>는 상위 5개 값을 나열한 표를 나타낸다. D_{repo}의 임계값을 지정하여 추가 레포지토리의 개수를 지정한다. 임계값을 낮추면 많은 레포지토리를 추가 대상으로 포함할 수 있기에 데이터의 양이 많아진다는 장점이 있지만 도메인의 유사성이 다소 떨어지는 레포지토리가 추가 대상으로 포함된다는 단점이 있다. 반대로 임계값을 높이면 유사한 도메인의 레포지토리가 추가 대상이 되지만 그만큼 레포지토리 수가 적어져 데이터의 수가 적어진다. 따라서 적절한 임계값을 지정할 필요가 있다. 본 연구에서는 D_{repo}의 임계값을 0.2로 지정하였고, Ruby/Ruby 레포지토리를 추가 레포지토리로 선정했다.

Table 2. Additional Repository List

Repository	C_{repo}	D_{repo}
Ruby/Ruby	129	0.2087
rubocop/rubocop	93	0.1504
rubygems/bundler	91	0.1472
rack/rack	84	0.1359
heartcombo/devise	75	0.1214

(2) 데이터 수집 및 네트워크 구성

추가 네트워크의 구성 과정은 기존 네트워크를 구성할 때와 같은 방식으로 진행된다. 우선 추가 레포지토리에 대한 풀-리퀘스트 정보를 수집한다. 총 5,326개의 풀-리퀘스트 정보에서 842명의 요청자, 137명의 배정 리뷰어, 662명의 리뷰어 정보를 수집했다. 이후 요청자와 리뷰어 간 연결 관계를 통해 동시발생 행렬을 생성한 후 추가 네트워크를 구성한다. 네트워크의 형태는 기존 네트워크와 동일하게 Directed graph이고 Weighted network이다. 노드 정보를 수집할 수 없는 1개의 노드와 중복된 노드를 제거하여 결과적으로 1,297개의 사용자 노드와 3,024개의 링크를 가지는 형태의 추가 네트워크가 구성된다.

4.3 확장 네트워크 구성

앞서 구성한 기존 네트워크와 확장 네트워크를 결합하는 방식을 통해 확장 네트워크를 구성한다. 두 네트워크에 모두 존재하는 요청자는 329명, 리뷰어는 631명으로 해당 노드들이 기존 네트워크와 추가 네트워크를 구조적으로 연결하는 역할을 한다. 확장 네트워크는 8,970개의 노드와 31,506개의 링크로 구성되어있고, 이는 기존 네트워크에 새로운 노드와 링크를 추가한 형태라고 볼 수 있다. 추후 링크예측 과정에서 새로운 링크가 등장할 때, 기존 네트워크의 노드와 새로운 노드 간의 연결을 기대해볼 수 있다. 위 과정을 거쳐 최종적으로 구성된 확장 네트워크를 GCN 모델을 통한 분석에 활용한다.

4.4 GCN 모델링

(1) 데이터 전처리

앞서 최종적으로 구성된 확장 네트워크의 데이터를 이용해 GCN 모델을 모델링한다. 31,506개의 링크를 Positive link로 활용하고, Negaitve sampling을 통해 Postive link와 같은 수, 즉 31,506개의 Negative link를 생성하여 활용한다. 이후 학습, 검정, 평가 데이터를 5:3:2의 비율로 분리한다. 결과적으로 학습 데이터는 각각 15,753개, 검정 데이터는 9,451개, 평가 데이터는 6,300개로 구성된다. 데이터 셋의 구성은 <Table 3>과 같다.

Table 3. Dataset used in this Paper

Data set	Positive link	Negative link	Total link
Train	15,753	15,753	31506
Valid	9,451	9,451	18,902
Test	6,300	6,300	12,600

Table 4. Model Settings

Data set	Positive link
Num of layer	2
Input layer	14
Hidden layer	16
Output dim	16
Optimizer	Adam
Learning rate	0.01
Epoch	100

(2) GCN 모델 구성

GCN 모델에 대한 설정 사항은 <Table 4>와 같다. 실험은 Pytorch 환경을 통해 수행하였으며, 두 개의 은닉층에 대한 활성화 함수는 ReLU, 출력층에는 Sigmoid 활성화 함수를 사용했다. 이후 연결된 노드 쌍 간의 유사도를 코사인 유사도로 측정하여 링크 예측을 수행한다. Optimizer는 Adam, 학습률은 최적화 결과 0.01, Epoch은 100회 진행하였다.

(3) GCN 모델 성능

GCN 모델 학습 후, 학습, 검정 데이터에 대해 성능을 평가한다. <Figure 5>와 <Figure 6>은 각각 Epoch별 손실(Loss) 변화 그래프, 정확도(Accuracy) 변화 그래프를 나타낸다. Table 5는 모델 각각 데이터에 대해 모델 성능 평가 지표를 보여준다. 모든 데이터에 대해 성능이 유사했으며, 정확도는 약 0.72, 정밀도는 0.66, 재현율은 0.91, F1 score는 0.76으로 준수한 성능을 기록했다.

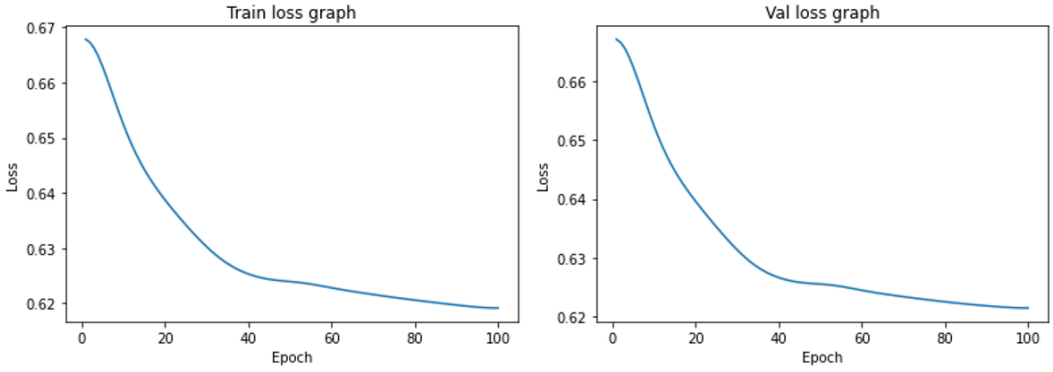

Figure 5. Loss Graph of Train, Validation Data

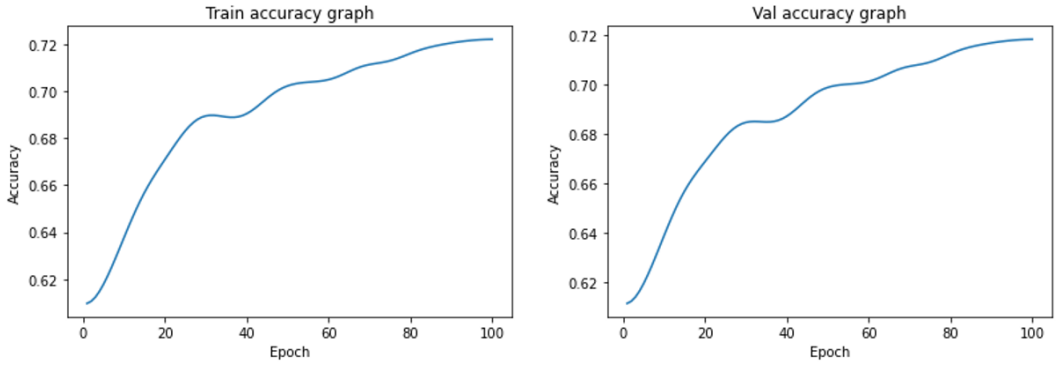

Figure 6. Accuracy Graph of Train, Validation Data

4.5 링크 예측 및 결과 분석

(1) 기존 네트워크와 비교 분석

본 연구의 기여는 확장 네트워크를 활용하여 GCN에 기반한 링크예측을 수행하였다는 점이다. 따라서 이를 검증하기 위해 기존 네트워크를 통해 학습하여 확장 네트워크에 대해 테스트한 결과와 성능을 비교하고자 한다. 기존 네트워크인 Rails/Rails 레포지토리의 데이터를 통해 모델링하고 확장 네트워크에 대해 테스트한 결과는 <Table 6>과 같다. 확장 네트워크를 통해 학습한 성능 평가 결과인 <Table 5>와 비교했을 때, 확장 네트워크가 약간 개선된 성능을 보임을 알 수 있다. 특히 기존 학습 모델은 확장 네트워크의 리뷰어를 추천하기에는 적합하지 않음을 알 수 있다. 이는 새로운 리뷰어를 추천해주기 위해 본 연구에서 도입한 네트워크 확장이 성능면으로도 준수함을 보여준다.

(2) 추천된 리뷰어 비교 분석

평가 데이터에 대해 링크 예측을 수행한 후 결과를 비교 분석한다. 평가 데이터는 Potsitive link와 Negative link가 각각 6,300개로 구성되어있다. 링크예측은 현재 연결되어있지 않은 링크 중 새롭게 연결될 가능성을 예측하는 기법이므로 6,300개의 Negative link를 통해 분석을 진행한다. 6,300개의 Negative link 중 리뷰어로부터 요청자에게 연결된 링크만을 필터링하여 총 2,268개의 링크만이 분석의 대상이 된다. 연결될 확률은 연결된 노드 간의 코사인 유사도를 통해 계산된다. 연결된 확률은 범위가 매우 넓으므로 임계값을 정해 일정 구간의 확률값만 필터링하여 사용하는 것이 바람직하다. 본 연구에서는 연결 확률이 상위 30%에 해당하는 681개의 링크를 분석에 활용한다.

681개의 링크를 링크에 연결된 노드 쌍의 정보를 통해 분류한다. 소스 노드인 리뷰어 정보와 타겟 노드의 요청자 정보에 대해 각각 네트워크 확장 이전에 어떤 네트워크에 포함되어있던 노드인지 출처를 파악한다. 노드의 출처는 기존 네트워크 혹은 추가 네트워크에만 포함된 경우와 두 네트워크 모두에 포함된 경우의 세 가지로 분류될 수 있다. <Table 7>은 681개 링크에 대해 연결된 노드쌍의 출처를 분류한 결과를 정리한 표다. 분류결과 네트워크 확장을 하지 않아도 가능한 연결, 즉 기존 네트워크 노드 간 연결(64.02%) 와 추가 네트워크 노드 간 연결(1.9%)가 약 66%로 대부분을 차지했다. 하지만 기존 네트워크의 요청자와 추가 네트워크의 리뷰어가 연결된 경우(4.85%), 추가 네트워크의 요청자와 기존 네트워크의 리뷰어가 연결된 경우(8.52%)와 같이 확장 후에만 나타날 수 있는, 완전히 새로운 리뷰어를 추천해주는 경우의 비율이 13%로 의미있는 수치를 기록했다.

Table 5. Model Performance

Data set	Accuracy	Precision	Recall	F1 score
Train	0.7224	0.6623	0.9070	0.7657
Validation	0.7187	0.6590	0.9062	0.7631
Test	0.7268	0.6667	0.9071	0.7685

Table 6. Model Performance of Existing Network

Data set	Accuracy	Precision	Recall	F1 score
Train	0.7043	0.6415	0.9259	0.7579
Validation	0.6932	0.6322	0.9237	0.7507
Test*	0.5104	0.5512	0.5823	0.5663

*Tested for extended network.

Table 7. Origin of Source and Target Node

Source Node (Reviewer)	Target Node (Puller)	Number of Connections	Ratio
Existing	Existing	436	64.02%
Additional	Additional	13	1.9%
Additional	Existing	33	4.85%
Existing	Additional	58	8.52%
Existing	Both	83	12.19%
Additional	Both	3	0.44%
Both	Existing	47	6.90%
Both	Additional	5	0.73%
Both	Both	3	0.44%

본 연구는 네트워크 확장이라는 새로운 개념을 통해 기존 네트워크의 요청자에게 새로운 리뷰어를 추천해주는 것을 목적으로 한다. 따라서 요청자가 기존 네트워크에 속하고, 리뷰어가 추가 네트워크에 속한 경우가 분석의 대상이 되고, 해당 리뷰어들을 잠재적 리뷰어 그룹으로 지정한다. 잠재적 리뷰어 그룹과 비교대상이 될 그룹은 요청자와 리뷰어가 모두 기존 네트워크에 속한 경우로 해당 리뷰어들은 기존 리뷰어 그룹으로 지정한다. 결과적으로 잠재적 리뷰어 그룹은 33명, 기존 리뷰어 그룹은 436명의 리뷰어로 구성된다.

기존 리뷰어 그룹과 잠재적 리뷰어 그룹에 대한 비교 분석을 수행하기 위해 이전에 수집된 사용자 지표인 Repository, Follower, Following, Commit, Main lanugage, Career 뿐 아니라 Text similarity를 사용한다. Repository, Follower, Following, Commit은 수치형 지표로 높을수록 활동력과 영향력이 높음을 나타낸다. 해당 지표는 지표별로 이상치를 제거한 후 정규성과 등분산성에 대한 검정 결과를 확인한 후, 비모수적 검정 방법인 맨-휘트니 U 검정(Mann-Whitney U test)을 통해 비교한다. Text simliarity는 요청자와 리뷰어 간 도메인의 유사성을 확인하기 위해 고려된 지표로 각 요청자, 리뷰어가 소유한 레포지토리 중 가장 많은 스타를 기록한 대표 레포지토리의 re-

Table 8. Basic Statistics for Each Group

	Existing reviewer group					Potential reviewer group				
	Repository	Follower	Following	Commit	Text Similarity	Repository	Follower	Following	Commit	Text Similarity
Avg	54.51	143.96	35.70	701.28	0.190260	79.12	59	29.09	559.91	0.184394
Std	57.05	886.80	98.80	1128.93	0.158102	77.87	85.07	73.84	902.45	0.137887
Q1	16	9	2	8	0.072943	19	8	0	27	0.088895
Median	37	27	10	248.5	0.158604	52	19	5	136	0.152825
Q3	73	70	34	948.25	0.279039	132	77	28	623	0.259211
Min	0	0	0	0	0	2	0	0	0	0
Max	503	16741	1457	11556	0.691450	284	416	409	3235	0.544043

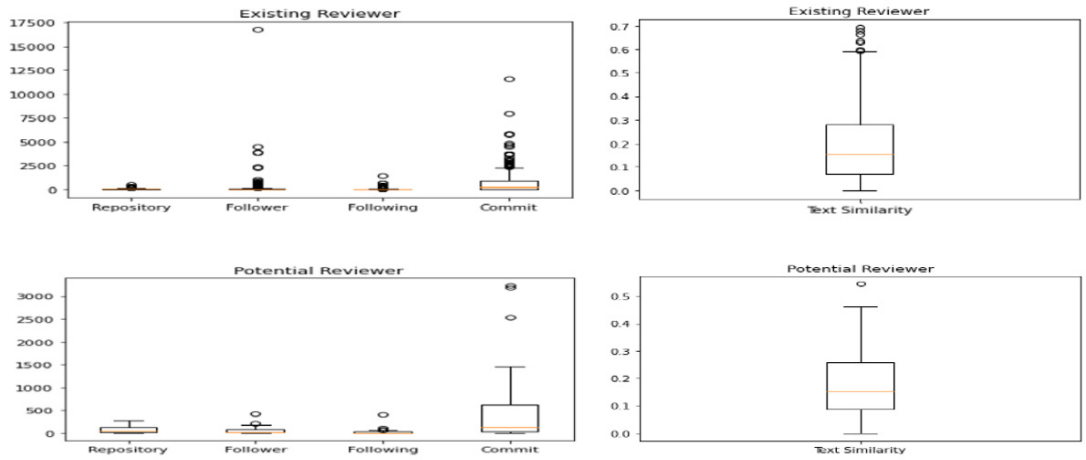

Figure 7. Data Distribution by Indicator

adme를 전처리 과정을 거쳐 TF-IDF(Term Frequency-Inverse Document Frequency) 벡터화한 후 이들의 코사인 유사도를 통해 측정한다. 이후 비모수적 검정 방법인 맨-휘트니 U 검정(Mann-Whitney U test)을 통해 비교한다. Main language는 요청자와 리뷰어의 주 프로그래밍 언어가 일치하는지 여부를 범주화한 후 카이제곱 검정을 시행하여 비교한다. Career는 가입일에 따라 Senior와 Junior로 범주화한 후 카이제곱 검정을 통해 비교한다.

<Table 8>은 각 지표 별 기초 통계를, <Figure 7>은 데이터의 분포를 나타낸다. 수치형 지표에 대해 $Q3+1.5\times IQR$ 이상의 값을 제외한 후 통계 분석에 사용한다. 하지만 이상치로 제외된 경우는 지표의 특성상 매우 높은 영향력을 가지는 리뷰어라고 설명할 수 있다. 따라서 이상치로 제외된 경우에 대해서도 통계를 확인했다. <Table 9>는 이상치 그룹의 통계표를 나타낸다. 그룹 별 이상치 비율은 크게 없으나, 샘플이 많은 기존 리뷰어 그룹이 평균치가 더 높음을 알 수 있다.

이후 각 지표별로 잠재적 리뷰어 그룹의 지표가 기존 리뷰어의 그룹보다 높은지를 판단하는 단측검정을 시행한다. Table 10은 두 그룹에 대한 통계적 차이를 확인하기 위한 단측검정 결과를 나타낸다.

분석결과 Repository 지표에 대해 P-value가 0.031929로 0.05보다 낮아 귀무가설을 기각, 잠재적 리뷰어 그룹이 기존 리뷰어보다 레포지토리 수가 많아 비교적 우수한 그룹임을 통계적으로 확인할 수 있다. 하지만 본 통계 분석의 결과는 두 그룹 간의 샘플 수가 매우 차이가 크다는 한계점이 있을 뿐 아니라 기존 리뷰어 그룹과 잠재적 리뷰어 그룹모두 표준편차가 매우 크기 때문에 통계적 유의성의 해석에 신중해야 한다. 지표들이 이상치를 제거했음에도 불구하고 우측으로 치우쳐져 있는 형태를 보이고, 편차가 크기 때문에, 단순히 비교했을 때는 평균치에 대해 차이가 있음에도 통계적으로는 차이가 있다고 판단하지 못한 것으로 보인다.

이종 프로젝트 참여자를 리뷰어로 추천하였을 때 우려되는 부분 중 하나는 추천된 리뷰어의 프로젝트 전문성 여부이다. 이를 반영하기 위한 Text similarity를 살펴보면 T-test 결과 단측검정이 기각되지 않았기 때문에 잠재 리뷰어 그룹이 기존 리뷰어 그룹보다 더 우수한 전문성을 가진다고 볼 수는 없다. 하지만 두 그룹의 유사도 평균값은 거의 비슷하거나 큰 차이가 나지 않았으며, 잠재적 리뷰어들이 동일 프로젝트 참여자 그룹이 아닌 것을 감안하면 잠재 리뷰어들의 프로젝트 유사성이 우려할 만큼 낮지는 않은 것으로 판단된다.

본 연구의 핵심 기여점이 네트워크 확장을 통해 보다 다양한 리뷰어를 추천하기 위한 것임을 감안할 때, 잠재 리뷰어 그룹이 기존 리뷰어 그룹의 Text simliarity 보다 프로젝트 전문성이 월등히 뛰어나지 않더라도 유사한 수준에서 더 많은 리뷰어를 추천할 수 있다는 점에서 장점을 가질 수 있다.

범주형 지표들의 그룹 간 통계분석의 결과는 <Table 11>에

Table 9. Basic Statistics after Outlier Removed

	Existing reviewer group					Potential reviewer group				
	Repository	Follower	Following	Commit	Text Similarity	Repository	Follower	Following	Commit	Text Similarity
Count	416	394	392	403	331	33	31	29	30	26
Avg	45.82	35.72	47.84	461.18	0.180120	79.12	42.61	9.58	317.13	0.170562
Std	38.46	36.67	48.68	575.49	0.145326	77.87	49.34	14.94	466.20	0.120002
Q1	15.75	8	1	6	0.072943	19	7	0	0	0.085461
Median	36	23	35	188	0.152518	52	18	15	58	0.148445
Q3	67	52	62	767.5	0.259773	132	71.5	64	360.25	0.245034
Min	0	0	0	0	0	2	0	0	0	0
Max	158	161	503	2290	0.589173	284	175	416	1459	0.462525

Table 10. T-test: One-sided Tests

	Existing reviewer group			Potential reviewer group			
	Avg	Std	Count	Avg	Std	Count	P-value
Repository	**45.82**	**38.46**	**416**	**79.12**	**77.87**	**33**	**0.031929***
Follower	35.72	36.67	394	59	85.07	31	0.536609
Following	47.84	48.68	392	29.09	73.84	29	0.976162
Commit	461.18	575.49	403	559.91	902.45	30	0.668602
Text Similarity	0.179477	0.143674	330	0.170562	0.120002	26	0.496445

$^{*}p \leqq 0.05$, $^{**}p \leqq 0.01$, $^{***}p \leqq 0.001$

Table 11. Chi-Square Test - Main Language & Career

Main language	Existing	Potential	χ^2	p	Career	Existing	Potential	χ^2	p
Same	233	18	0	1.0	Senior	303	18	2.52	0.11
Different	203	15			Junior	133	15		

서 확인할 수 있다. 범주형 변수는 이상치가 존재하지 않아 전체 샘플에 대해 카이제곱 검정을 진행했다. 두 지표 모두 P-value가 0.05보다 크기 때문에 그룹 간 차이가 없는 것으로 나타났다. 이 중 요청자와 리뷰어 간 콘텐츠 유사성으로 활용된 지표인 Main Language의 경우 P-value의 값이 1로 두 그룹이 완벽히 유사함을 나타냈다. 따라서 기존 리뷰어와 잠재적 리뷰어가 요청자에 대해 콘텐츠 유사성이 비슷하다고 할 수 있고, 콘텐츠 유사성 관점에 있어 잠재적 리뷰어의 추천은 적합하다고 해석할 수 있다.

5. 결론 및 한계점

본 연구는 네트워크 확장이라는 개념을 통해 완전히 새로운 코드 리뷰어를 추천하는 프레임워크를 제안했다는 의의가 있다. 기존 연구의 추천시스템에서는 리뷰어가 레포지토리의 소유자 및 지인으로 리뷰어 풀이 한정되어있다는 한계가 존재했다. 그러나 다양한 리뷰어를 추천해준다면 다양한 의견을 수렴할 수 있다. 따라서 완전히 새로울 뿐만 아니라 코드 리뷰 역량과 유사한 도메인 지식을 갖춘 새로운 리뷰어를 추천해주기 위하여 네트워크 확장이라는 과정을 거쳤다. 기존 네트워크의 리뷰어 정보를 통해 유사한 도메인의 추가 레포지토리를 선정한 후, 이에 대한 네트워크를 구성 및 결합하여 확장 네트워크를 구성함을 통해 새로운 리뷰어를 추천했다. 결과적으로 기존 방식에서는 추천될 수 없던 새로운 양질의 리뷰어를 추천해주었다는 의의가 있다. 또한 본 연구에서는 GCN 모델을 활용하여 이전 연구에서 중요하게 고려되었던 콘텐츠 유사성 기반 지표와 사용자 간 연결관계를 동시에 활용했다는 의의가 있다. 리뷰어 추천에 있어 콘텐츠의 유사성은 주요한 요소이고, 협업이 잦은 깃허브의 특성상 연결관계 또한 충분히 고려될 가치가 있는 요소이다. 본 연구에서는 이 두 가지 요소를 노드 정보와 연결 관계를 GCN 모델을 사용하여 동시에 반영했다.

그러나 몇가지 한계점도 존재한다. 첫째로 추가 레포지토리 선정 과정의 임계값 조정이다. 추가 레포지토리를 선정하는 과정에서 임계값을 변화함에 따라 추가 대상으로 선정될 레포지토리의 수가 달라지고 이는 성능과 추천 결과에 영향을 미친다. 따라서 추후 연구로 임계값 변화에 따른 민감도 분석이 요구된다. 둘째로 분석 대상이 될 레포지토리가 대규모 프로젝트에만 한정되어있다는 점이다. 본 연구에서는 많은 양의 데이터를 활용하기 위해 기존 연구에서 사용된 대규모 프로젝트인 Rails/Rails 레포지토리를 사용했다. 많은 풀-리퀘스트 정보로 인해 요청자와 리뷰어 간 연결이 잦았고, 다양한 리뷰어 정보를 통해 추가 대상으로 선정될 수 있는 레포지토리도 많아졌다. 하지만 기존 레포지토리의 규모가 작은 경우에는 적은 리뷰어 정보로 인해 추가 대상이 될 레포지토리 수도 적어지고, 추가 네트워크 구성을 통한 원활한 추천을 기대하기 어렵다. 따라서 소규모 레포지토리에 대해서도 검증이 요구된다. 마지막으로 완전히 새로운 리뷰어를 추천했기 때문에, 리뷰어의 적합성에 대한 명확한 평가가 어렵다는 점이다. 본 연구에서는 잠재적 리뷰어와 기존 리뷰어를 통계분석을 통해 비교했으나, 그룹 간 샘플 수의 차이가 커 유의미한 비교는 어려웠다. 더 큰 규모의 추가 네트워크를 사용하여 샘플 수의 차이를 줄인 후 비교 분석한다면 유의미한 분석이 가능할 것이다. 특히 새로 추천된 리뷰어의 프로젝트 전문성을 파악할 수 있는 지표로 본 연구에서는 프로젝트 유사성을 활용하였으나, 유사성 이외에도 신규 리뷰어의 전문성을 나타낼 수 있는 다양한 지표를 추가하여 비교한다면 보다 의미 있는 분석이 될 수 있을 것이다.

참고문헌

Ai, B., Qin, Z., Shen, W., and Li, Y. (2022), Structure Enhanced Graph Neural Networks for Link Prediction. arXiv preprint arXiv:2201.05293.

Balachandran, V. (2013), Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In *2013 35th International Conference on Software Engineering (ICSE)* (pp. 931-940). IEEE.

Cannistraci, C. V., Alanis-Lobato, G., and Ravasi, T. (2013), From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks, *Scientific Reports*, **3**(1), 1613.

Çetin, H. A., Doğan, E., and Tüzün, E. (2021), A review of code reviewer recommendation studies: Challenges and future directions, *Science of Computer Programming*, **208**, 102652.

Chatziasimidis, F. and Stamelos, I. (2015), Data collection and analysis of GitHub repositories and users, In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, IEEE, 1-6.

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012), Social coding in GitHub: transparency and collaboration in an open software repository, In *Proceedings of the ACM 2012 conference on computer supported cooperative work*, 1277-1286.

Hasan, M. A. and Zaki, M. J. (2011), A survey of link prediction in social networks, *Social Network Data Analytics*, 243-275.

Islam, M. K., Aridhi, S., and Smail-Tabbone, M. (2020), A comparative study of similarity-based and GNN-based link prediction approaches, arXiv preprint arXiv:2008.08879.

Jeong, G., Kim, S., Zimmermann, T., and Yi, K. (2009), Improving

code review by predicting reviewers and acceptance of patches, *Research on software analysis for error-free computing center Tech-Memo (ROSAEC MEMO 2009-006)*, 1-18.

Jiang, J., Lo, D., Zheng, J., Xia, X., Yang, Y., and Zhang, L. (2019), Who should make decision on this pull request? Analyzing time-decaying relationships and file similarities for integrator prediction, *Journal of Systems and Software*, **154**, 196-210.

Kononenko, O., Rose, T., Baysal, O., Godfrey, M., Theisen, D., and De Water, B. (2018), Studying pull request merges: a case study of shopify's active merchant, In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, 124-133.

Kumar, A., Singh, S. S., Singh, K., and Biswas, B. (2020), Link prediction techniques, applications, and performance: A survey, *Physica A: Statistical Mechanics and its Applications*, **553**, 124289.

Liao, Z., Wu, Z., Li, Y., Zhang, Y., Fan, X., and Wu, J. (2020), Core-reviewer recommendation based on Pull Request topic model and collaborator social network, *Soft Computing*, **24**, 5683-5693.

Lü, L. and Zhou, T. (2011), Link prediction in complex networks: A survey, *Physica A: Statistical Mechanics and its Applications*, *390*(6), 1150-1170.

Su, Z., Zheng, X., Ai, J., Shen, Y., and Zhang, X. (2020), Link prediction in recommender systems based on vector similarity, *Physica A: Statistical Mechanics and its Applications*, **560**, 125154.

Thongtanunam, P., Tantithamthavorn, C., Kula, R. G., Yoshida, N., Iida, H., and Matsumoto, K. I. (2015), Who should review my code? a file location-based code-reviewer recommendation approach for modern code review, In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, IEEE, 141-150.

Wessel, M., Serebrenik, A., Wiese, I., Steinmacher, I., and Gerosa, M. A. (2020), What to expect from code review bots on GitHub? A survey with OSS maintainers, In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, 457-462.

Yang, C., Zhang, X. H., Zeng, L. B., Fan, Q., Wang, T., Yu, Y., Yin, G. and Wang, H. M. (2018), RevRec: A two-layer reviewer recommendation algorithm in pull-based development model, *Journal of Central South University*, **25**(5), 1129-1143.

Ying, H., Chen, L., Liang, T., and Wu, J. (2016), Earec: leveraging expertise and authority for pull-request reviewer recommendation in github, In *Proceedings of the 3rd International Workshop on CrowdSourcing in Software Engineering*, 29-35.

Yu, Y., Wang, H., Filkov, V., Devanbu, P., and Vasilescu, B. (2015), Wait for it: Determinants of pull request evaluation latency on github, In *2015 IEEE/ACM 12th working conference on mining software repositories*, IEEE, 367-371.

Yu, Y., Wang, H., Yin, G., and Ling, C. X. (2014), Reviewer recommender of pull-requests in GitHub, In *2014 IEEE International Conference on Software Maintenance and Evolution*, IEEE, 609-612.

Yu, Y., Wang, H., Yin, G., and Wang, T. (2016), Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?, *Information and Software Technology*, **74**, 204-218.

Zhang, X., Yu, Y., Gousios, G., and Rastogi, A. (2022), Pull request decisions explained: An empirical overview, *IEEE Transactions on Software Engineering*, **49**(2), 849-871.

저자소개

전병민 : 서울과학기술대학교 산업공학과에서 학사학위를 취득하고, 학석사연계과정을 통해 서울과학기술대학교 데이터사이언스학과에 진학하여 2022년 석사학위를 취득하였다. 주요 관심 연구분야는 데이터마이닝, 머신러닝,데이터 기반 비즈니스 어낼리틱스 등이다.

금영정 : KAIST 산업공학과에서 학사학위를 취득하고, 삼성 SDS 정보전략실을 거쳐 서울대학교에서 석박사 통합과정으로 2007년에 박사학위를 취득하였다. 현재 서울과학기술대학교 산업공학과 및 데이터사이언스학과에서 부교수로 재직 중이다. 주요 연구분야는 데이터 사이언스를 바탕으로 기술경영 현장을 분석하기 위한 비즈니스 어낼리틱스 기법을 개발하는 것으로, 유망기술 발굴, 비즈니스 혁신, 특허분석 등을 연구하고 있다.