Cross-Project Defect Regression을 위한 전이 학습 기법 적용성 연구

최정환 [O†] 류덕산[‡]

[†]연세대학교 인공지능학과 [‡]전북대학교 소프트웨어공학과

jeongwhan.choi@yonsei.ac.kr, duksan.ryu@jbnu.ac.kr

A Study on the Applicability of Transfer Learning Techniques for Cross-Project Defect Regression

Jeongwhan Choi [O†] Duksan Ryu[‡]

[†]Department of Artificial Intelligence, Yonsei University

[‡]Department of Software Engineering, Jeonbuk National University

요 약

교차 프로젝트 환경에서의 소프트웨어 결함 예측 (Cross-Project Defect Prediction, CPDP)은 Cold-start 문제를 다루기에 중요하다. 하지만 결함 유무를 예측하는 것보다 결함 수에 대해 예측하는 것이 리소스 할당에 더 효과적이다. 현재 Cross-Project Defect Regression(CPDR) 연구는 적은 편이며, 최근 연구들은 이 문제를 랭킹 문제로 접근하고 있다. 본 연구는 CPDP가 아닌 CPDR 환경을 고려하여 결함 개수를 예측하는데 어떤 기법을 통해 성능 향상을 할 수 있을 지 연구한다. CPDP에서 주로 사용되는 전이 학습(Transfer Learning) 기법이 CPDR 환경에서의 적용 가능성이 있는 지 확인한다. 그리고 CPDR 환경에서의 Learning to Rank 방법의 효과를 확인한다. CPDP에서 주로 사용되는 전이 학습 기법이 CPDR 환경에서의 효과가 적음을 보이며 Learning to Rank 방법이 큰 효과를 보인다는 것을 보인다. 이러한 연구 결과를 통해 랭킹 문제로 CPDR을 접근하는 것이 결함 수에 따른 인스펙션 할 모듈들을 제공하는데 효과적이라 기대한다.

1. 서론

소프트웨어 결함 연구는 최근 가장 활발한 소프트웨어 엔지니어링 연구 분야 중 하나이다. 소프트웨어 결함 예측은 실제 버그 레코드와 함께 실제 소프트웨어 데이터를 활용하여 기계학습 모델을 훈련시킨다. 결함이 발생하기 쉬운 모델들을 효과적으로 찾음으로써 품질 보장과 효과적인 리소스 할당을 할 수 있다.

이미 릴리즈 된 프로젝트의 경우 WP(Within-Project) 환경에서 여러 버전의 프로젝트들을 포함하고 있으며 충분한 데이터를 가지고 있을 수 있다. 그렇지만 프로젝트의 초기 단계에서는 모델을 훈련하기 위한 충분한 데이터가 제공되지 않기 때문에, CPDP가 해결책이 된다. 최근까지 나온 CPDP를 위한 연구들은 결함 유무에 대한 예측을 연구하지만, 결함 개수에 따른 연구는 CP(Cross-Project) 환경에서는 많지 않다. 결함 수를 파악하는 연구는 결함 수에 따른 인스펙션 할 모듈을 제공해줄 수 있으므로 효과적인 리소스 할당에 도움을 줄 수 있다. CP 환경에서 이를 예측하는 연구를 CPDR(Cross-Project Defect Regression)이라고 한다. CPDR을 위한 회귀 모델(regressor)로 Ridge, Lasso Regressor, ElasticNet를 사용한다. 그리고 기존 CPDP에서 주로 사용되는 전이 학습(Transfer Learning) 기법을 CPDR에 적용하여 효과를 확인한다.

‡ 교신 저자; 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2019R1G1A1005047)

CPDR에서의 Regression는 랭킹 기법으로 확장이 가능하다. 결함 수를 파악하는 연구는 인스펙션할 모듈을 순위별로 학습하고 제안할 수 있어야하기 때문이다. 그렇기에 LTR(Learning to Rank) 기법인 RankNet을 사용하여 실험한다. 정보 검색 분야의 필수 접근 방식 중 하나로 알려진 LTR은 이제 추천 시스템 및 정보 검색에 대한 순위 모델을 구축하기 위해 빠르게 발전하고 있다. 하지만 아직 소프트웨어 결함 예측 분야에서는 적용된 연구들은 적다.

본 논문은 이러한 CPDR에 CPDP의 전이 학습과 LTR에 대한 적용 가능성과 성능 향상에 대한 효과를 분석한다. 본 연구를 통해 전이 학습의 효과가 적다는 것을 효과크기를 통해 보인다. 그리고 RankNet을 Regressor 대신 사용할 때, 큰 효과크기를 보이며 RankNet이 CPDR 환경에서 적합하다는 것을 보인다. 본 연구가 기여하는 바는 Regressor들과 LTR 모델들을 비교한 CPDR 연구는 우리가 아는 한 처음인 점이다. 그리고 본 연구 결과를 통해, 후속 연구자들에게 CPDR 연구 방향을 가이드 할 수 있을 것으로 기대한다.

2. 관련연구

소프트웨어 결함 수 예측 방법은 최근 들어 연구가 활발하게 나타나고 있다[1]–[4]. Rathore et al.은 결함 수 예측에 대해 경험적인 연구를 제안하였지만 CPDR 환경을 고려한 연구는 아니다[5]. CPDR 환경에서 Yang et al.은 LTR 기법을 제안하고

최근에는 Lasso와 Ridge를 이용하여 Cross-Version 환경에서 연구를 하였다[2][4]. 이는 우리의 연구와 같이 CPDR 환경이 아닌 Cross-Version 환경에서 연구를 진행하였다. Wang et al.은 LTR 기법들에 대해서 비교 실험을 하고 RankNet을 기반으로 불균형 문제를 해결하고자 한 모델을 제안하였다[6].

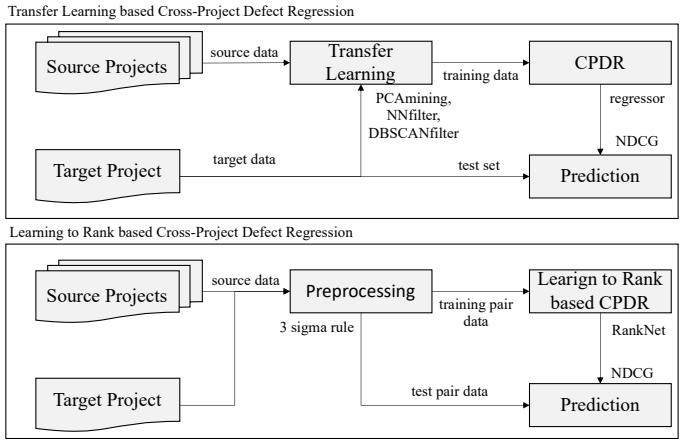

그림1. 전이 학습과 LTR 기반 CPDR 프레임워크

3. 연구 방법

본 연구에서는 CPDR 환경에서 전이 학습 기법의 적용가능성을 확인한다. 그림1의 상단처럼 전이 학습 기법을 사용한 방법과 그렇지 않은 CPDR 방법을 비교한다. PCAmining, NNfilter, DBSCANfilter를 전이 학습으로 사용한다[7]–[9]. 전이 학습은 다대일 관계의 소스 프로젝트, 타겟 프로젝트 상에서 진행된다. 예를 들어 타겟 프로젝트 ant의 초기 버전을 예측하기 위해 다른 프로젝트들이 소스 프로젝트로써 사용된다. Regressor는 Lasso, Ridge와 ElasticNet을 이용하여 CPDR 환경에서 결함 수를 예측한다.

하단처럼 LTR 기반 CPDR 방법을 LTR을 사용하지 않은 기법과 비교한다. LTR 모델은 RankNet을 사용한다. RankNet은 Burges et al.에 의해 제안되었고 실제 랭킹 문제를 해결하는데 유용하다는 것이 입증되었다[10]. LTR 모델에 적용하기 위해 기존 데이터 전처리가 필요하다. RankNet 모델은 pairwise 기반 모델이기 때문이다. 이를 위해 결함 개수에 대한 평균과 분산을 계산한 후 매핑 규칙을 정의하여 관련성을 정의한다. 3-sigma를 기반으로 한 매핑 규칙은 식(1)과 같다. r_i이 0일 경우는 결함 개수가 0이며 결함과의 관련성이 없고, 3일 경우에는 결함에 대한 관련성이 높다는 것을 의미한다.

$$r_i = \begin{cases} 0, & x_i = 0 \\ 1, & x_i \in (0, \hat{\mu} + \hat{\sigma}] \\ 2, & x_i \in (\hat{\mu} + \hat{\sigma}, \hat{\mu} + 2\hat{\sigma}] \\ 3, & x_i \in (\hat{\mu} + 2\hat{\sigma}, +\infty] \end{cases} \quad (1)$$

4. 실험 설계

4.1 연구 질문

CPDR 환경에서 전이 학습의 효과와 LTR 방법의 효과를 확인하기 위해 통계적 검정을 이용하여 각 연구질문들의 귀무가설을 기각할 수 있는지 확인한다.

RQ1. CPDR 환경에서 전이 학습을 적용하는 방법이 이를 적용하지 않는 것보다 효과가 있는가?

- *H_0: 전이 학습을 적용한 모델의 성능이 적용하지 않은 모델들의 성능과 유사하다.*
- *H_A: 전이 학습을 적용한 모델의 성능이 적용하지 않은 모델들의 성능과 다르다.*

RQ2. Learning to Rank 모델은 Regressor 모델보다 CPDR 환경에서 성능 향상에 대한 효과가 있는가?

- *H_0: CPDR 환경에서 Learning to Rank 모델은 Regressor 모델의 성능과 유사하다.*
- *H_A: CPDR 환경에서 Learning to Rank 모델은 Regressor 모델의 성능과 다르다.*

4.2 데이터셋

본 연구는 결함 개수를 포함하고 있는 데이터가 필요하기에 MORPH 데이터셋을 사용한다[11]. 20개의 소프트웨어 정적 메트릭으로 구성되어 있으며 표1은 각 프로젝트의 릴리즈 버전 수와 결함 수에 대한 범위를 보인다.

표1. 프로젝트 데이터셋 정보

Project	ant	camel	jedit	log4j	lucene
# of releases	5	4	5	3	3
# range of defects	[0,10]	[0,47]	[0,45]	[0,10]	[0,47]

4.3 평가 척도

평가 척도로 Recall, Precision등을 사용할 수 도 있지만, 상대적인 위치 정보도 고려하는 NDCG(Normalized Discounted Cumulative Gain) 측면에서 순위 결과를 평가한다. 식(2)에서는 r은 i번째 위치에서 결과의 등급별 관련성이고 Z_k는 정규화 요소이다. 본 연구에서는 k가 5와 10인 측면에서 평가한다.

$$NDCG(k) = Z_k \cdot \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log_2(i+1)} \quad (2)$$

5. 실험 결과

5.1 RQ1에 대한 실험 결과

세 모델에 대해 CPDR를 진행한 실험 결과는 표2와 같으며 ElasticNet이 모든 프로젝트에 대해 평균 NDCG10이 0.46으로 가장 좋은 결과를 보인다.

CPDP에서 사용되는 전이 학습 기법들인 PCAmining, NNfilter, DBSCANfilter를 사용한 결과는 표3과 같다. 표3은 전이 학습을 적용하였을 때와 그렇지 않을 때의 NDCG 성능을 효과크기로 나타내었다. Ridge와 PCAmining를 사용하면 성능이 향상되었다는 것을 효과크기를 통해 확인할 수 있다. 다른 조합의 경우에는 전이 학습을 사용하였을 때의 NDCG 성능 차이가 전혀 없다. Ridge의 경우 NDCG5의 효과크기가 0.2이기에 작은 효과가 있다는 것을 확인할 수 있다. Ridge를 사용한다면 RQ1의 귀무가설을 약하게 기각할 수 있지만, 실험을 통해 전이 학습은 성능 향상 효과가 없음을 확인할 수 있다.

표2. Regressor 모델들에 대한 실험 결과

model	Lasso		Ridge		ElasticNet	
target	NDCG5	NDCG10	NDCG5	NDCG10	NDCG5	NDCG10
ant	0.0524	0.1431	0.0524	0.156	0.0524	0.1377
camel	0.2021	0.1528	0.1275	0.1406	0.1275	0.0964
jedit	0.6309	0.6309	0.6309	0.6309	0.6309	0.6309
log4j	0.9584	0.9548	0.9582	0.9502	0.9584	0.9557
lucene	0.43	0.4311	0.0008	0.3557	0.499	0.5002

표3. PCAmining, NNfilter, DBSCANfilter 기법을 적용한 Regressor들의 성능 차이에 대한 Cohen's D 효과크기 결과

Regressor	Metrics	PCAmining	NNfilter	DBSCANfilter
Lasso	NDCG5	-0.038	-0.5882	-1.7132
	NDCG10	0.0019	-0.5796	-1.7323
Ridge	NDCG5	**0.218**	-0.1672	-0.2075
	NDCG10	0.0501	-0.3818	-0.5205
ElasticNet	NDCG5	-0.035	-0.3224	-1.6393
	NDCG10	-0.007	-0.2796	-1.6664

5.2 RQ2에 대한 실험 결과

표4를 통해 RankNet을 사용한 경우 NDCG5와 NDCG10에 대한 평가척도 모두 1에 가까운 Cohen's D 효과크기를 보인다. 효과크기는 세 가지 Regressor에 대한 실험 결과에 대해 계산하였다. Cohen's D를 통해 RQ2에서의 귀무가설을 기각할 수 있으며 대립가설인 "CPDR 환경에서 Learning to Rank 모델은 Regressor 모델의 성능과 다르다."를 채택할 수 있다. 실험 결과를 통해서도 NCDG 평가척도의 크기가 더 큰 것을 확인할 수 있다.

표4. RankNet의 실험 결과와 각 Regressor들에 대한 효과크기

Model		RankNet	
Target Project		NDCG5	NDCG10
ant		0.8429	0.9296
camel		0.5958	0.6944
jedit		0.587	0.5872
log4j		0.8832	0.8196
lucene		0.6255	0.6728
Cohen's D	Lasso	1.0167	1.1203
	Ridge	1.1719	1.22
	ElasticNet	0.953	1.1203

6. 위협 요소

외부 유효성에 대한 위협은 5 개의 프로젝트(20 개 버전)만을 사용한 것이다. 이 위협을 완화시키기 위해 데이터셋을 추가하여 실험할 계획이다. 내부 유효성에 대한 위협은 NDCG 만을 사용한 것이 내부 유효성에 대한 위협이므로 향후 다른 성능 메트릭을 추가할 계획이다.

7. 결론 및 향후 연구

본 연구는 CPDP 가 아닌 CPDR 환경을 고려하여 결함 개수를 예측하는데 어떤 기법을 통해 성능 향상을 할 수 있을지 연구한다. CPDP 에서 주로 사용되는 전이 학습 기법보다는 LTR 방법이 결함 개수를 예측하고 순위를 계산하는데 적합하다는 것을 연구질문들을 통해 확인할 수 있었다. RQ1 와 달리 RQ2 에서는 큰 효과크기를 통해 RankNet 을 이용한 성능 향상을 확인하였다.

향후 연구로는 클래스 불균형 문제에 대한 연구와 추가적인 LTR 모델들을 적용할 계획이다.

참고 문헌

[1] M. Chen and Y. Ma, "An empirical study on predicting defect numbers," in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, 2015.

[2] X. Yang, K. Tang, and X. Yao, "A learning-to-rank approach to software defect prediction," *IEEE Trans. Reliab.*, vol. 64, 2015.

[3] M. Nevendra and P. Singh, "Software bug count prediction via AdaBoost.R-ET," *Proc. 2019 IEEE 9th Int. Conf. Adv. Comput. IACC 2019*, pp. 7–12, 2019.

[4] X. Yang and W. Wen, "Ridge and Lasso Regression Models for Cross-Version Defect Prediction," *IEEE Trans. Reliab.*, vol. 67, no. 3, pp. 885–896, Sep. 2018.

[5] S. S. Rathore and S. Kumar, "An empirical study of some software fault prediction techniques for the number of faults prediction," *Soft Comput.*, 2017.

[6] F. Wang, J. Huang, and Y. Ma, "A Top-k Learning to Rank Approach to Cross-Project Software Defect Prediction," *Proc. - Asia-Pacific Softw. Eng.*, vol. 2018-Decem, 2018.

[7] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in *Proceedings - International Conference on Software Engineering*, 2006.

[8] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empir. Softw. Eng.*, vol. 14, no. 5, Oct. 2009.

[9] K. Kawata, S. Amasaki, and T. Yokogawa, "Improving relevancy filter methods for cross-project defect prediction," in *Proceedings - International Conference on Applied Computing and Information Technology and International Conference on Computational Science and Intelligence, ACIT-CSI 2015*, 2015.

[10] C. Burges *et al.*, "Learning to rank using gradient descent," in *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[11] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *ACM International Conference Proceeding Series*, 2010.