자동차 소프트웨어 교차-프로젝트 결함 예측을 위한 전이 학습 모델 비교 연구

최정환 ○, 류덕산, 김순태
전북대학교 소프트웨어공학과
{jeongwhan.choi, duksan.ryu, stkim}@jbnu.ac.kr

Comparative Study of Transfer Learning Models for Cross-Project Automotive Software Defect Prediction

Jeongwhan Choi○, Duksan Ryu, Suntae Kim
Department of Software Engineering, Jeonbuk National University, South Korea

요 약

대부분의 소프트웨어 결함 예측 접근법은 동일한 프로젝트의 데이터에 대해 훈련되고 적용된다. 그러나 종종 새 프로젝트에는 충분한 훈련 데이터가 없다. 이때 교차-프로젝트 결함 예측 기법을 활용할 수 있다. 이 연구에서는 자동차 소프트웨어 프로젝트 데이터를 사용하여 교차-프로젝트 결함 예측을 위한 전이 학습(Transfer Learning) 기법들을 실험하여 비교한다. 기존 전이 학습 기법들인 TCA, Burak Filter, DBSCAN Filter들을 사용한다. 그리고 결함 예측 모델을 사용할 때 랜덤 선택에 비해 얼마나 코드 인스펙션 비용이 감소하는지 분석한다. 통계적 검정을 통해 DBSCAN Filter 또는 TCA 기법을 이용할 때 Burak Filter 보다 높은 예측 성능을 보이며, 더 많은 코드 인스펙션 노력을 감소할 수 있다는 것을 보인다. 이러한 연구 결과로 자동차 도메인에서, 신규 소프트웨어 프로젝트에 대한 효과적인 품질 보증 활동 수행을 지원할 것으로 기대된다.

1. 서론

소프트웨어 결함 예측에 대한 접근 방법은 대부분 동일 프로젝트 내의 데이터로 학습되고 적용된다. 결함 예측 모델이 충분한 양의 데이터로 학습된다면 예측 성능이 높다는 것은 이미 잘 알려져있다[1]. 하지만 새로운 프로젝트를 개발하는 경우, 학습을 위한 과거 데이터가 없기 때문에 결함 예측을 할 수 없다. 전이 학습 기법[2]을 활용함으로써 CPDP(Cross-Project Defect Prediction)[3]는 학습 데이터 부족을 다루는 효과적인 방법으로 점차 대중화 되고있다[4]. 일반적인 아이디어는 다른 프로젝트(소스 도메인 프로젝트)의 데이터를 활용하여 모델을 구축하고 이를 적용하여 현재 모델(타겟 도메인 프로젝트)의 결함을 예측하는 것이다.

Broy[5]가 수행 한 분석에 따르면 자동차 소프트웨어는 개발예산의 최대 40%를 소비한다. 자동차 소프트웨어는 최대 1,000만 개의 LOC(Lines of Code)를 포함할 수 있다. 이 산업에서 테스트는 모든 개발 단계의 핵심이고 결함 예측은 개발 프로세스에 도움을 주는 도구이다.

이 논문은 Altinger et al.[6]이 제시한 자동차 소프트웨어 프로젝트에서 얻은 소스코드 정적 메트릭 데이터를 사용한다. CPDP를 고려하며 이를 위해 전이 학습 기법들을 비교 실험한 후 코드 인스펙션 노력 비용에 대해 분석한다. 기존 전이 학습 기법들인 TCA(Transfer Component Analysis)와 Burak Filter, DBSCAN Filter들을 사용한다. 분류 모델로는 Balanced Random Forest을 사용하고 결함 예측 모델의 코드 인스펙션 노력 감소 비율에 대해 분석한다. 통계적 검정을 통해 DBSCAN Filter, TCA를 이용한 전이 학습 기법이 높은 예측 성능과 FIR(File Inspection Reduction)과 LIR(LOC Inspection Reduction)이 높다는 것을 보인다.

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단(No. NRF-2019R1G1A1005047)과 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업(IITP-2020-2017-0-01628)의 지원으로 수행된 연구임.

2. 관련 연구

CPDP는 문헌조사 연구에서 입증하듯이 주목을 받고 있다[7],[8]. 인스턴스 선택은 CPDP에 대해 학습된 모델을 전이하는 가장 일반적인 방법이며 타겟 인스턴스와 유사한 소스 인스턴스가 모델을 학습하도록 선택한다[9]–[12]. Turhan et al.[10]이 제안한 Burak Filter는 유클리드 거리를 기준으로 타겟 데이터의 각 인스턴스에 대한 교차 프로젝트 데이터에서 k 개의 근접 이웃을 선택한다. Kawata et al.의 논문에서 사용한 DBSCAN Filter 방법은 타겟, 교차 프로젝트 데이터를 결합하여 DBSCAN으로 하위 클러스터를 찾고 그 클러스터 내에서 관련 있는 데이터를 선택한다. Nam et al.[13]은 TCA[9] 방법을 CPDP에 성공적으로 적용했다. TCA는 소스 프로젝트를 선택하는 대신, 소스와 타겟 프로젝트의 분포가 가까운 공유 잠재 공간(Shared Latent Space)에 직접 매핑한다. 본 연구에서는 관련 연구의 방법들인 TCA와 Burak Filter, DBSCAN Filter를 사용한다. 자동차 소프트웨어 프로젝트에서의 결함 예측은 Altinger et al.의 데이터셋을 기반으로 연구가 진행되고 있다[6], [14]–[16].

3. 접근 방법

신규 프로젝트의 결함 예측을 위해서 CPDP를 사용한다. 하지만, 소스와 타겟 프로젝트간 데이터셋 분포가 다르기 때문에, 이를 해결하기 위해서 전이 학습 기법을 활용한다. 소프트웨어 메트릭 정보에 대한 범위가 각각 다르기에 이에 대해서 최소-최대 정규화를 진행한다. 타겟 프로젝트를 제외한 프로젝트들은 소스 프로젝트로 하나 또는 두 개로 구성되어 실험한다. 전이 학습을 통해 타겟 프로젝트와 관련된 데이터로 예측 모델을 학습한다. 모델은 Balanced Random Forest를 사용하고 타겟 프로젝트에 대해 결함을 예측하고 이에 대한 코드 인스펙션 감소 비율을 측정하여 분석한다.

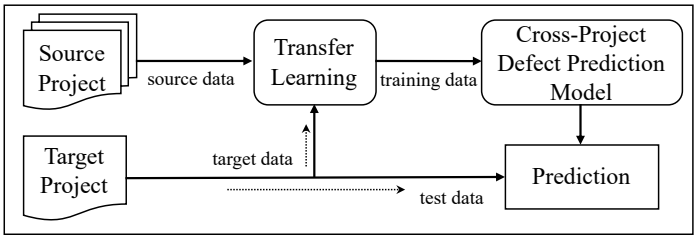

그림 1. 접근 방법

4. 실험 및 결과

4.1 연구 질문

RQ1 *자동차 소프트웨어의 교차 프로젝트 결함 예측에서 가장 좋은 성능을 보이는 전이 학습 기법은 무엇인가?*

- H_0: 모든 전이 학습 기법의 성능이 유사하다.
- H_A: 전이 학습 기법 간에 성능의 차이가 있다.

TCA의 차원은 5, Burak Filter의 이웃 수는 3, DBSCAN Filter는 10으로 실험한다. 분류 모델로는 Balanced Random Forest을 사용하여 불균형한 클래스 문제를 다룬다. 이는 각 부트스트랩 샘플을 무작위로 언더-샘플링하여 클래스의 균형을 조정한다. 데이터셋은 불균형한 클래스를 가지고 있으므로 이에 적합한 성능 척도로 ROC-AUC를 사용한다. 이는 ROC 곡선 아래에 해당되는 면적이며 각 축은 PD(Probability of Detection)과 PF(Probability of False Alarm)으로 구성된다.

RQ2 *교차-프로젝트 결함 예측 모델에서 가장 코드 인스펙션 노력을 감소시키는 전이 학습 기법은 무엇인가?*

- H_0: 모든 전이 학습 기법의 코드 인스펙션 노력 감소 비율은 유사하다.
- H_A: 전이 학습 기법 간에 코드 인스펙션 노력 감소 비율의 차이가 있다.

예측 모델이 어떻게 코드 인스펙션에 대한 노력을 감소하는지 보인다. 랜덤 선택 인스펙션에 비교하여 얼마나 감소하는지 나타내기 위한 척도는 FIR와 LIR이다[17].

FIR (File Inspection Reduction). FIR은 랜덤 선택과 비교하여 예측 모델을 사용하여 인스펙션 할 때, 동일한 PD(Probability of Detection)를 달성하기 위해 줄어든 인스펙션 할 파일 수의 비율이다. FI(File Inspection)는 전체 파일에 대해 인스펙션 할 파일 수의 비율이고 FIR은 이와 같이 정의한다: FIR = (PD – FI) / PD

LIR (LOC Inspection Reduction). LIR은 랜덤 선택과 비교하여 예측 모델을 사용하여 인스펙션 할 때, 동일한 PrD(Predictive Defectiveness)를 달성하기 위해 줄어든 LOC의 비율이다. LI(LOC Inspection)는 전체 LOC에 대해서 인스펙션 할 LOC의 비율이며 LIR은 다음과 같이 정의한다: LIR = (PrD – LI) / PrD

4.2 실험환경

데이터셋은 표1과 같으며 세 가지의 프로젝트에서 추출한 정적 소프트웨어 메트릭 정보(LOC, McCabe, Halstead)들이 포함되어 있다. 실험은 Python 3.7 환경에서 진행한다.

표1. Project A, K, L에 대한 데이터 정보

Project	A	K	L
# of Files	1,908	2,515	2,891
# of Bug	77	112	61

4.3 RQ1: 전이 학습 기법 비교 실험 결과

모든 전이 학습 기법의 성능을 일원분산분석의 효과 크기(Effect Size)를 이용해 분석한 결과 Cohen's f는 1.014이다. 이는 0.4보다 크기에 영향이 크다는 것을 알 수 있고 대립 가설을 채택할 수 있다. 표2에서 알 수 있듯이 Project A가 타겟 데이터일 경우 TCA 기법을 사용할 때 ROC-AUC가 0.7928로 가장 높은 성능을 보인다. Project L의 경우, 나머지 두 프로젝트가 소스 데이터이고 TCA 기법을 이용할 때 0.96의 성능을 보인다. 전체 실험에서의 평균값으로는 DBSCAN Filter 기법을 이용할 때 좋은 예측 성능을 보이는 것을 알 수 있다.

표2. 각 전이 학습 기법에 대한 교차-프로젝트 결함 예측 모델 성능 비교

Source Project	Target Project	TCA	Burak Filter	DBSCAN Filter
		ROC-AUC		
K,L	A	0.7312	0.66	0.6854
K		**0.7928**	**0.6785**	0.6774
L		0.5613	0.5434	**0.6985**
A,L	K	0.607	**0.7892**	**0.7729**
A		0.6944	0.7218	0.759
L		**0.7463**	0.7154	0.7268
A,K	L	**0.96**	0.4784	0.8058
A		0.9156	0.509	0.917
K		0.8662	**0.6127**	**0.9515**
Mean		0.764	0.634	0.777

4.4 RQ2: 코드 인스펙션 감소 측정 결과

표3에서는 FIR와 LIR에 대한 인스펙션 감소 측정 결과를 보여준다. RQ1에서 성능 척도가 높은 소스 데이터를 사용하여 측정하였다. Project L의 경우 DBSCAN Filter를 이용할 경우 파일은 91.4%, LOC는 33.6%만큼 인스펙션 노력이 감소될 수

있다는 것을 보인다. 각 프로젝트에 대해서 평균값을 계산한 결과 Burak Filter보다 TCA 또는 DBSCAN 전이 기법을 사용하는 것이 인스펙션 노력을 줄일 수 있다는 것을 알 수 있다. 그리고 각 전이 학습 기법에 대한 FIR과 LIR의 통계적 검정 결과 FIR는 Cohen's f가 0.696, LIR은 0.716이다. 이는 모두 0.4보다 크기에 대립 가설을 채택할 수 있다.

표3. 각 전이 학습 기법에 대한 FIR, LIR 측정 결과

	TCA		Burak Filter		DBSCAN Filter	
Target Project	FIR	LIR	FIR	LIR	FIR	LIR
A	**0.794**	**0.094**	0.622	-0.747	0.793	-0.536
K	0.837	**0.162**	0.838	0.015	**0.846**	0.153
L	0.901	0.32	0.792	-0.209	**0.914**	**0.336**
Mean	0.844	0.192	0.751	-0.314	0.851	-0.016

5. 위협 요소

불균형한 클래스 데이터는 구성 유효에 대한 위협이다. 이 위협을 줄이기 위해 불균형한 클래스에 적합한 ROC- AUC를 성능 평가 척도로 선택하였다. 외부 유효성에 대한 위협은 3개의 프로젝트만을 사용한 것이다. 현재 자동차 도메인에서 가용한 결함 데이터셋은 이들 3개가 전부이며, 추가적인 데이터셋이 확보되면, 제안하는 기법을 적용할 계획이다.

6. 결론 및 향후 연구

이 연구는 CPDP를 고려하며 이를 위해 전이 학습 기법들을 비교 실험한 후 코드 인스펙션 노력에 대해 분석한다. 전이 학습 기법들과 분류 모델인 Balanced Random Forest을 사용한다. RQ1, RQ2에서 TCA, DBSCAN Filter 기법을 사용하는 것이 다른 기법보다 더 나은 예측 성능을 보이고 노력 감소 비율이 우수하다는 것을 통계적 검정을 통해 보인다. 이는 자동차 산업의 개발 프로세스에서의 노력 비용을 줄이는데 도움을 줄 수 있다. 향후 연구로는 머신러닝 알고리즘과 전이 모델에 대한 하이퍼파라미터 최적화 연구를 진행할 계획이다. 또한, 소프트웨어 변경 메트릭을 추가로 사용하여 연구할 계획이다.

참고 문헌

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*. 2012.

[2] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*. 2010.

[3] L. C. Briand, W. L. Melo, and J. Wüst, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE Trans. Softw. Eng.*, 2002.

[4] Y. Zhou *et al.*, "How far we have progressed in the journey? An examination of cross-project defect prediction," *ACM Trans. Softw. Eng. Methodol.*, 2018.

[5] A. Pretschner, M. Broy, I. H. Krüger, and T. Stauner, "Software Engineering for Automotive Systems : A Roadmap," 2007.

[6] H. Altinger, S. Siegl, Y. Dajsuren, and F. Wotawa, "A novel industry grade dataset for fault prediction based on model-driven developed automotive embedded software," in *IEEE International Working Conference on Mining Software Repositories*, vol. 2015-Augus, pp. 494–497, 2015.

[7] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Transactions on Software Engineering*. 2019.

[8] S. Herbold, A. Trautsch, and J. Grabowski, "A Comparative Study to Benchmark Cross-Project Defect Prediction Approaches," *IEEE Trans. Softw. Eng.*, 2018.

[9] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Networks*, 2011.

[10] 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.*, 2009.

[11] K. Kawata, S. Amasaki, and T. Yokogawa, "Improving relevancy filter methods for cross-project defect prediction," in *Proceedings - 3rd International Conference on Applied Computing and Information Technology and 2nd International Conference on Computational Science and Intelligence, ACIT-CSI 2015*, 2015.

[12] D. Ryu, J. I. Jang, and J. Baik, "A Hybrid Instance Selection Using Nearest-Neighbor for Cross-Project Defect Prediction," *J. Comput. Sci. Technol.*, 2015.

[13] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proceedings - International Conference on Software Engineering*, 2013.

[14] H. Altinger, S. Herbold, F. Schneemann, J. Grabowski, and F. Wotawa, "Performance tuning for automotive Software Fault Prediction," in *SANER 2017 - 24th IEEE International Conference on Software Analysis, Evolution, and Reengineering*, pp. 526–530, 2017.

[15] H. Altinger, S. Herbold, J. Grabowski, and F. Wotawa, "Novel insights on cross project fault prediction applied to automotive software," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9447, pp. 141–157, 2015.

[16] H. Altinger, Y. Dajsuren, S. Siegl, J. J. Vinju, and F. Wotawa, "On Error-Class Distribution in Automotive Model-Based Software," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, pp. 688–692, 2016.

[17] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, "Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities," *IEEE Trans. Softw. Eng.*, vol. 37, no. 6, pp. 772–787, 2011.