API들의 동시 출현 그래프와 커뮤니티 구성용 Louvain 방법을 활용한 안드로이드 악성 앱 탐지

WDSC2025

운영체제 및 보안 연구실

노경민, 이승민, 김유담, 안석현, 조성제

INDEX

01 서론

문제 배경

악성 앱 탐지의 어려움

❖ 개념 드리프트: 시간이 지남에 따라 데이터 분포가 바뀌는 현상

❖ 개념 드리프트 때문에 악성 앱의 사용 패턴이 바뀌어 악성 앱 탐지에 어려움이 발생

- 탐지 기법을 우회하기 위해 악성 앱이 지속적으로 진화 → 개념 드리프트 발생
- 기존의 연구는 주기적 재학습이나 드리프트 인지를 통해 이에 대응
- 주기적 재학습 없이도 개념 드리프트에 대응하여 성능이 높게 나오는 방법이 필요

관련 연구

정적 분석 기반 악성 앱 탐지

권한, API 호출, 문자열 기반

전통적 머신 러닝 [4]
딥러닝 [7]

API 존재 여부 등의 개별 정보에만 의존하여 의미 관계나 구조적 상호작용 고려 X

그래프 구조 활용 탐지 기법

API 호출 시퀀스 기반

마르코프 체인 [12]
호출 그래프 [13]
GCN (Graph Convolutional Network)

그래프 생성 과정이 복잡하고 그래프 구조 변화에 취약, 고정된 시점의 그래프 구조에만 초점

개념 변화 탐지 및 적응 연구

드리프트 탐지 기법

재학습 기반 [17]
분포 변화 감지 모델 업데이트 [18]
스트리밍 방식의 적응형 학습

지속적인 데이터 수집과 주기적 재학습 등 높은 운용 비용이 존재

[4] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, “Drebin: Effective and explainable detection of android malware in your pocket,” Proceedings of the Network and Distributed System Security Symposium (NDSS), pp. 23–26, 2014.
[7] S. Kim, S. Kim, S. Lee, and H. Kim, “MAPAS: A practical deep learning-based android malware detection system,” International Journal of Information Security, vol. 21, pp. 217–233, 2022.
[12] M. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, “MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models,” Proceedings of the Network and Distributed System Security Symposium (NDSS), 2017.
[13] Y. Wang, J. Zhang, and Y. Wang, “Android Malware Detection Method Based on Graph Convolutional Networks,” Proceedings of the IEEE 6th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), pp. 1–6, 2024.
[17] S. Park, H. Lee, D. Kim, H. J. Moon, S. Cho, and Y. Hwang, “Enhancing the Sustainability of Machine Learning-Based Malware Detection Techniques for Android Applications,” IEEE Transactions on Information Forensics and Security, vol. 20, no. 1, 2025.
[18] J. Zhang, Y. Wang, and X. Liu, “Adaptive Android Malware Detection via Distribution Change Detection and Model Update,” IEEE Transactions on Information Forensics and Security, vol. 18, pp. 1234 1245, 2023.

연구 목표

개념 드리프트를 재학습 없이 해결하는 것이 목표

우리는 이 개념 드리프트를 재학습 없이 다루는 정적 특징 정보 (API 호출) 기반 악성 앱 탐지 체계를 제안한다.

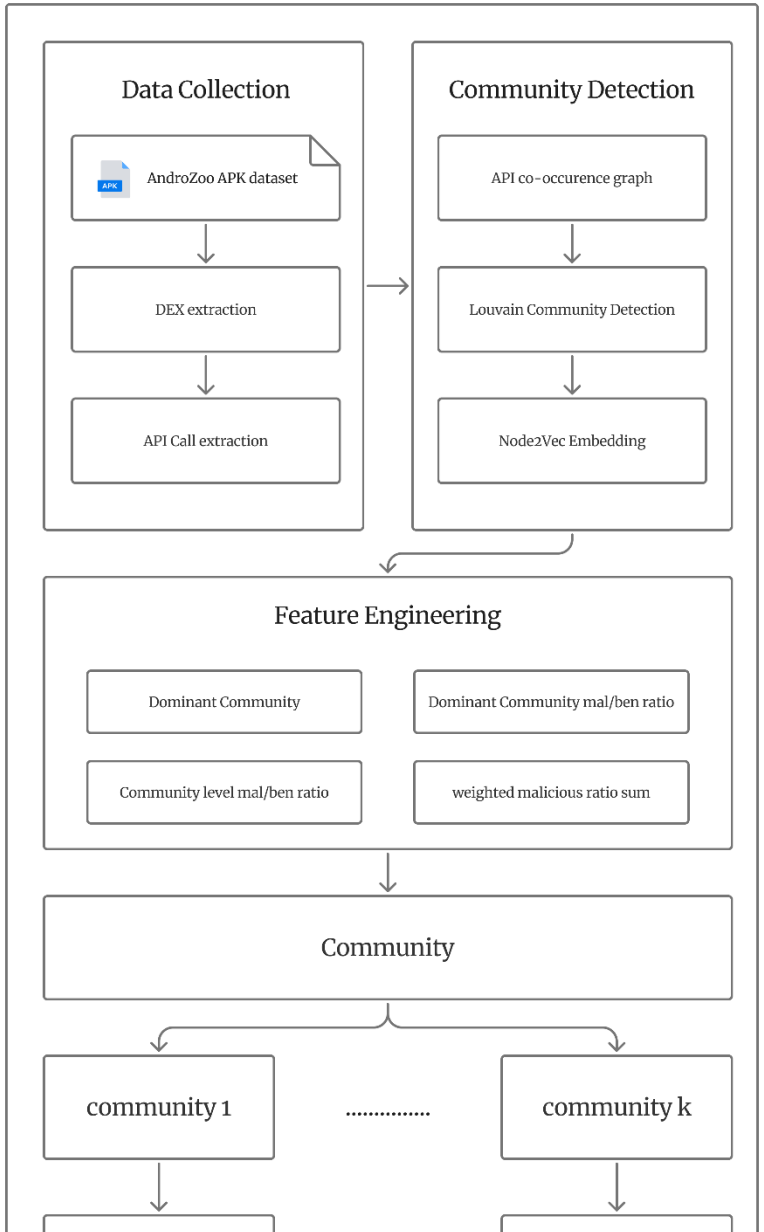
어떻게 해결하는가?

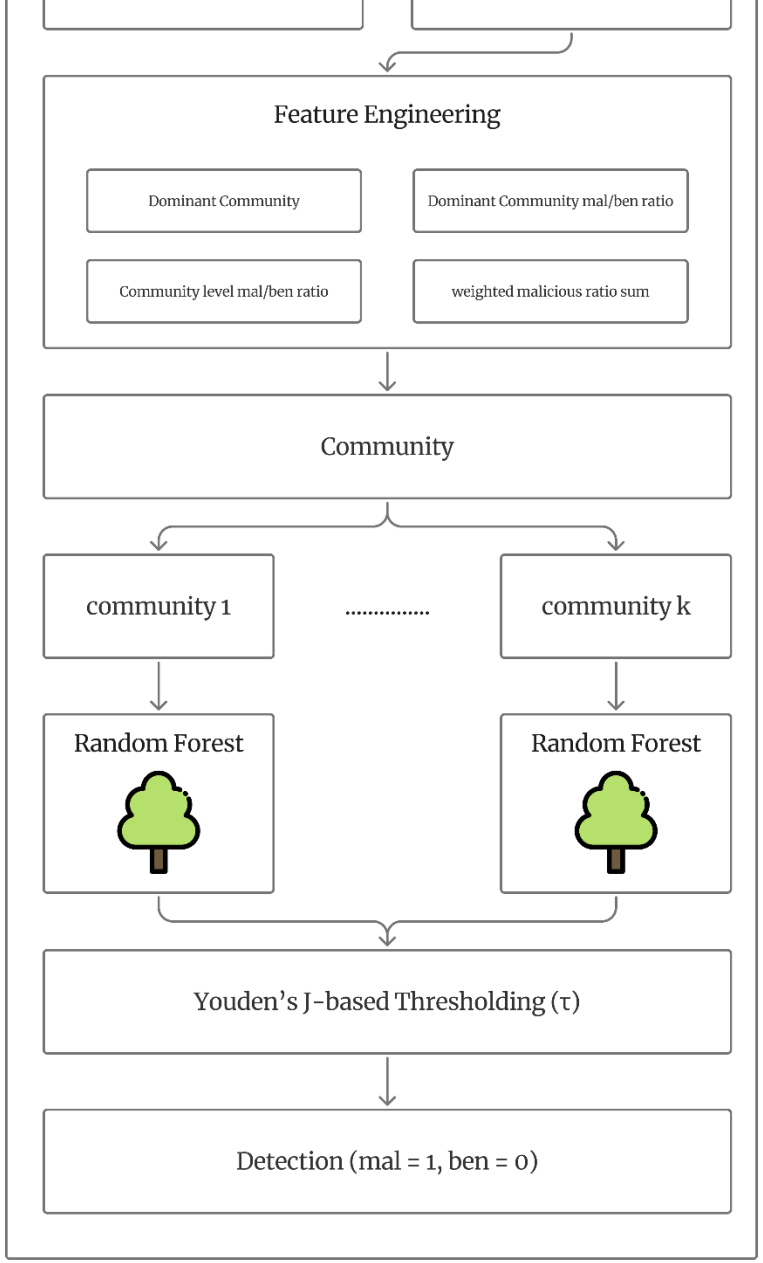
1. **동시 출현 그래프 생성**: 학습 데이터에서 API 동시 출현 그래프를 생성
2. **클러스터링 수행**: Louvain 알고리즘으로 의미 기반 커뮤니티로 클러스터링 수행
3. **지역 의사결정**: 커뮤니티별 분류기와 임계값 설정으로 지역 분포 차이를 반영

- API 간의 의미 관계와 구조적 상호작용을 반영 → 동시 출현 그래프 생성
- 동시 출현 그래프 기반 Louvain 알고리즘을 통한 커뮤니티 구성 → 그래프 구조 변화에 강건
- 낮은 운용 비용을 가지고 재학습이 없음에도 강건한 성능 유지

02 제안 프레임워크

Overall Framework





Data Feature Extraction

❖ 데이터셋 구성

- AndroZoo에서 가져온 76,000개 안드로이드 APK 파일이 원천
- APK 파일에서 DEX 파일을 추출하여 얻은 1,848개의 API 호출 frequency 특징을 갖는 입력 정보 구성
- 학습 데이터는 2014-2017년까지, 테스트 데이터는 2018-2023년까지의 데이터를 사용
- 학습 데이터와 테스트 데이터를 엄격하게 분리하여 Data Leakage를 차단

Year	Benign	Malicious	**Year**	Benign	Malicious
2014	5,000	5,000	2019	3,000	3,000
2015	5,000	5,000	2020	3,000	3,000
2016	5,000	5,000	2021	3,000	3,000
2017	5,000	5,000	2022	3,000	3,000
2018	3,000	3,000	2023	3,000	3,000
Total	23,000	23,000	**Total**	15,000	15,000

Community Detection

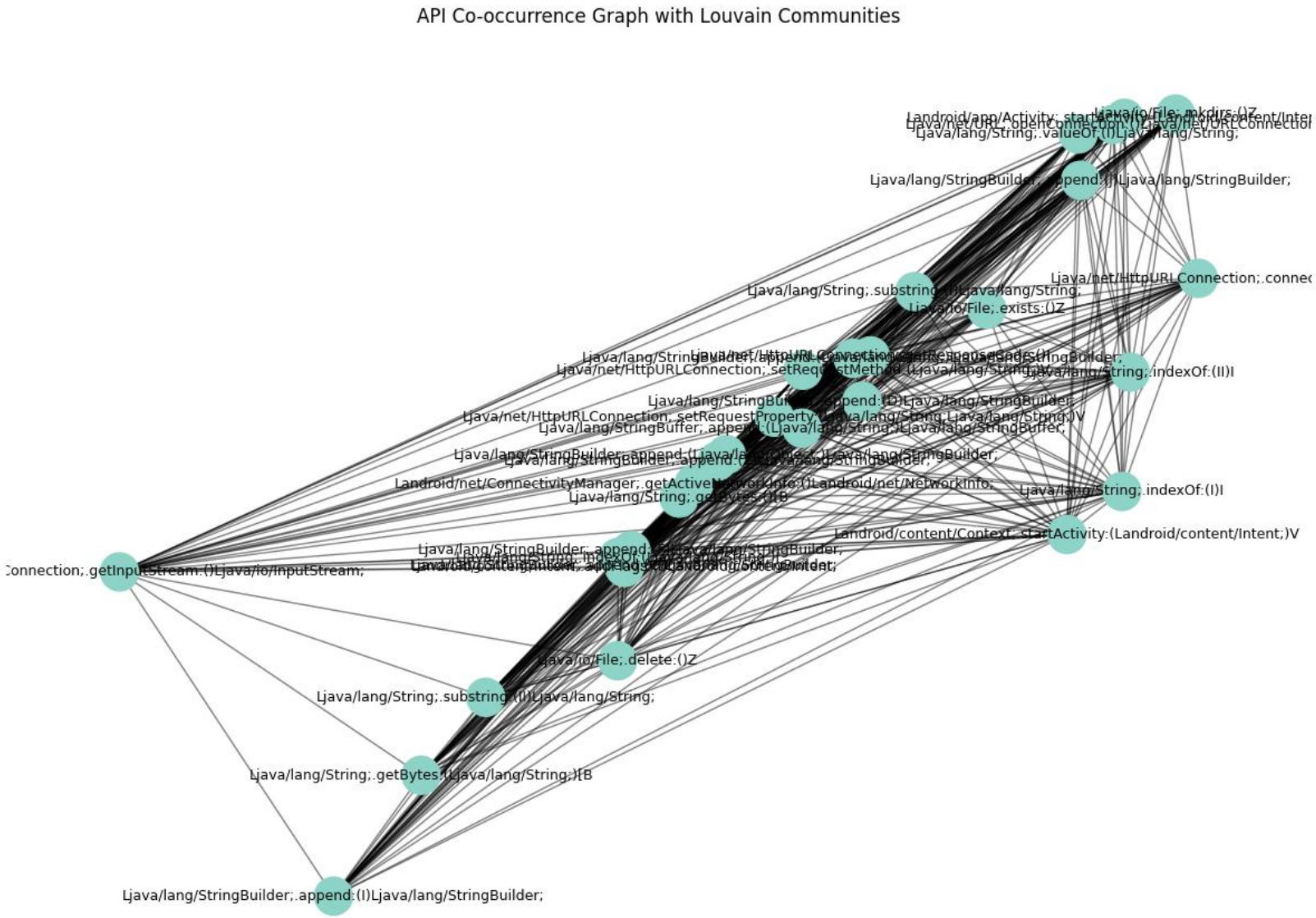
❖ 커뮤니티 구성

- SNA (Social Network Analysis)
  - 대상 간 관계로 이루어진 네트워크의 구조적 특성을 분석 → **커뮤니티와 중요 노드를 식별**
  - 보통 사회학 연구에서 사회 구성체의 SNS 활동, 인터넷 밈 확산, 정보 순환 등의 연구에 활용

- 본 연구에서의 활용법
  - 사회 구성체 → APK, 사회 구성체의 활동 → API 호출 이라고 가정
  - SNA 중, Louvain 알고리즘을 사용하여 API 호출 동시 출현 그래프를 분석
  - **노드는 API 호출, 엣지는 동일 앱 내 동시 출현**을 의미

Community Detection

❖ 커뮤니티 구성



Community Detection

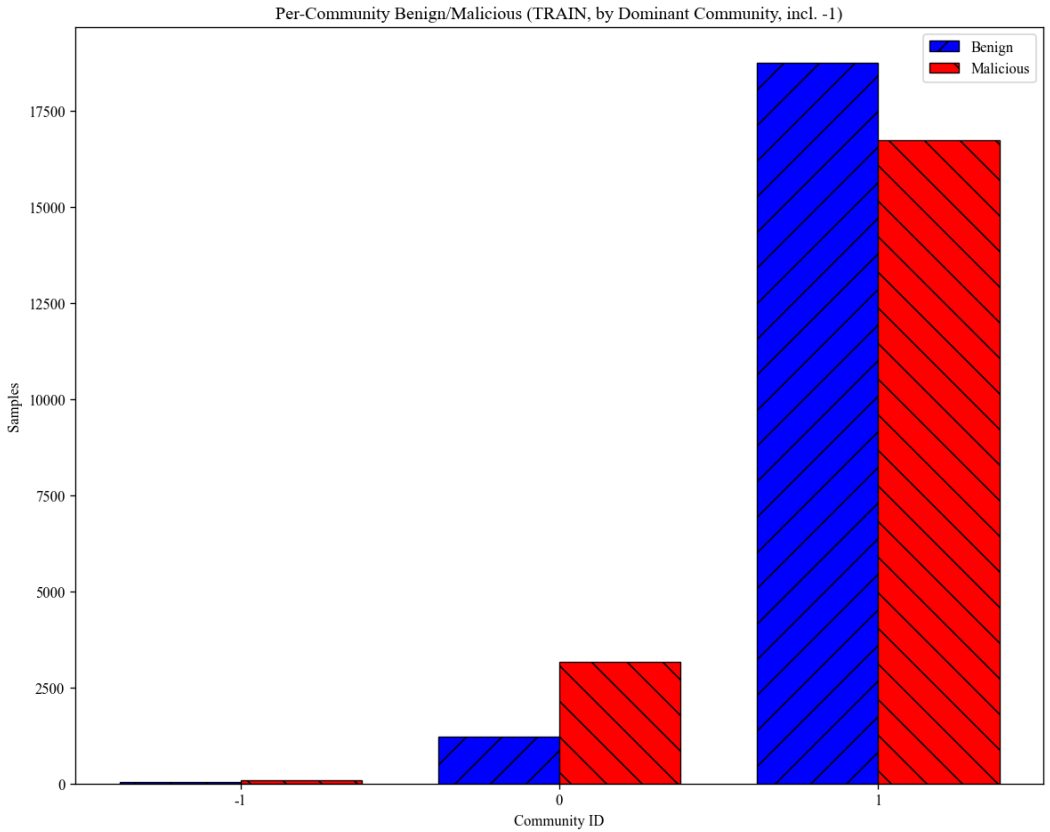
❖ 커뮤니티 구성

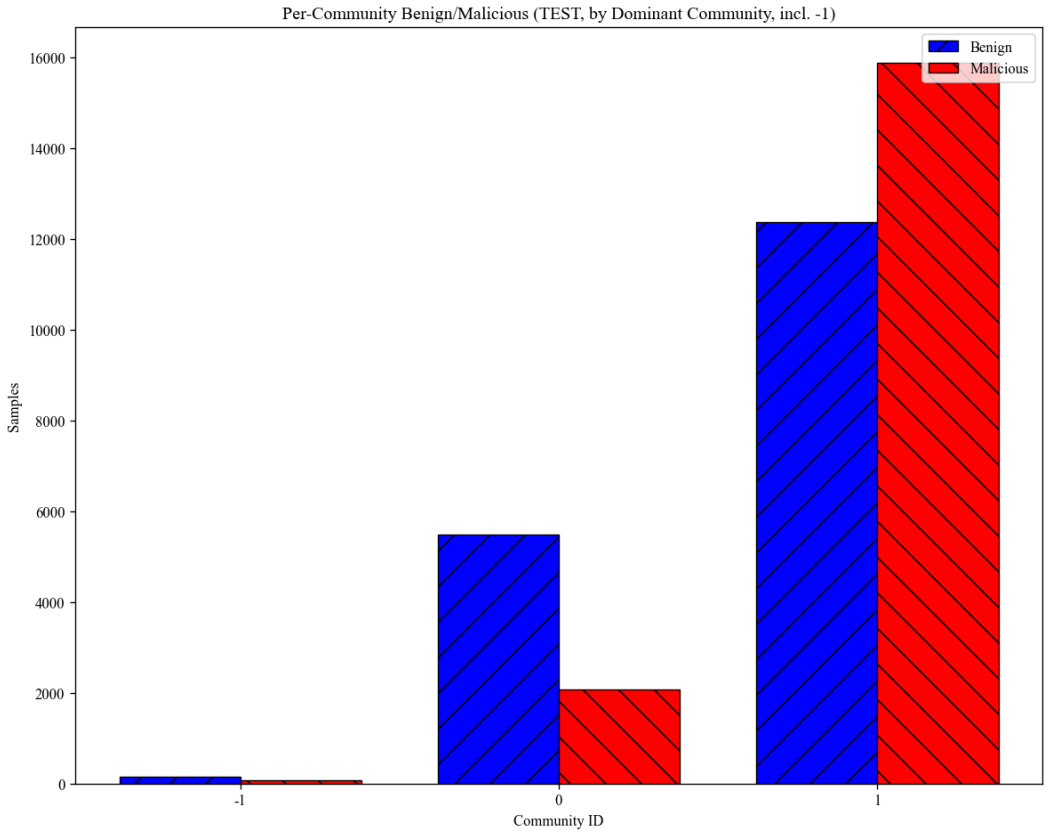
- **커뮤니티 구성의 순서**
  - 동시 출현 그래프 G 구성 → Louvain 알고리즘 적용 → 커뮤니티 구조 rnt
  - G가 구성되면, 그래프의 **구조적 응집성**을 분석하는 것이 필요: Louvian 적용

- **Louvain 알고리즘의 역할?**
  - Louvain 알고리즘은 그래프의 모듈성을 최대화하는 방향으로 노드들을 **클러스터링**
  - 각 커뮤니티는 동일한 기능적 맥락 내에서 자주 함께 호출되는 API들의 집합으로 해석

Community Detection

❖ 커뮤니티 구성 결과





Community Detection

❖ 우세 커뮤니티 할당

- Louvain 알고리즘: 그래프의 모듈성을 최대화하는 방향으로 노드들을 클러스터링
- 각 API 노드는 하나의 커뮤니티에 속함
- 각 앱은 자신이 호출한 API들 중 가장 많이 속한 커뮤니티를 우세 커뮤니티로 할당

→ 각 앱의 주요 기능적 중심축을 규정, 이후 특징 생성 및 탐지기의 학습 단계에서 기준 역할 수행

Feature Engineering

❖ 커뮤니티 기반 특징 정보 생성

이름	유형	생성 기준	비고
dominant_comm_mal	스칼라	우세 커뮤니티의 악성 앱 비율	훈련 기간 데이터 참조
dominant_comm_ben	스칼라	우세 커뮤니티의 정상 앱 비율	훈련 기간 데이터 참조
MR_mean	스칼라	호출된 API들의 악성 비율 평균	훈련 기간 데이터 참조
embedding	벡터	호출 API 임베딩의 평균 벡터	Node2Vec 기반

→ 스칼라 3개 + 벡터 8개로 총 11개 특징 정보가 커뮤니티 기반으로 추출되어 앱당 추가됨.

Model Training & Decision Making

❖ 커뮤니티별 분류기 학습

- 각 앱은 우세 커뮤니티 ID에 따라 하나의 커뮤니티에 소속
- 각 커뮤니티마다 독립적인 Random Forest 분류기를 학습 → 커뮤니티별 행위 특성에 최적화된 탐지 가능

❖ 임계값 선택

- 학습 데이터의 예측 확률로 ROC 곡선을 계산
- Youden's J 통계량이 최대가 되는 지점 = 최적 임계값
- 민감도와 특이도의 균형 기반 → 클래스 불균형 상황에서도 안정적인 기준점 제공

03 결과 및 분석

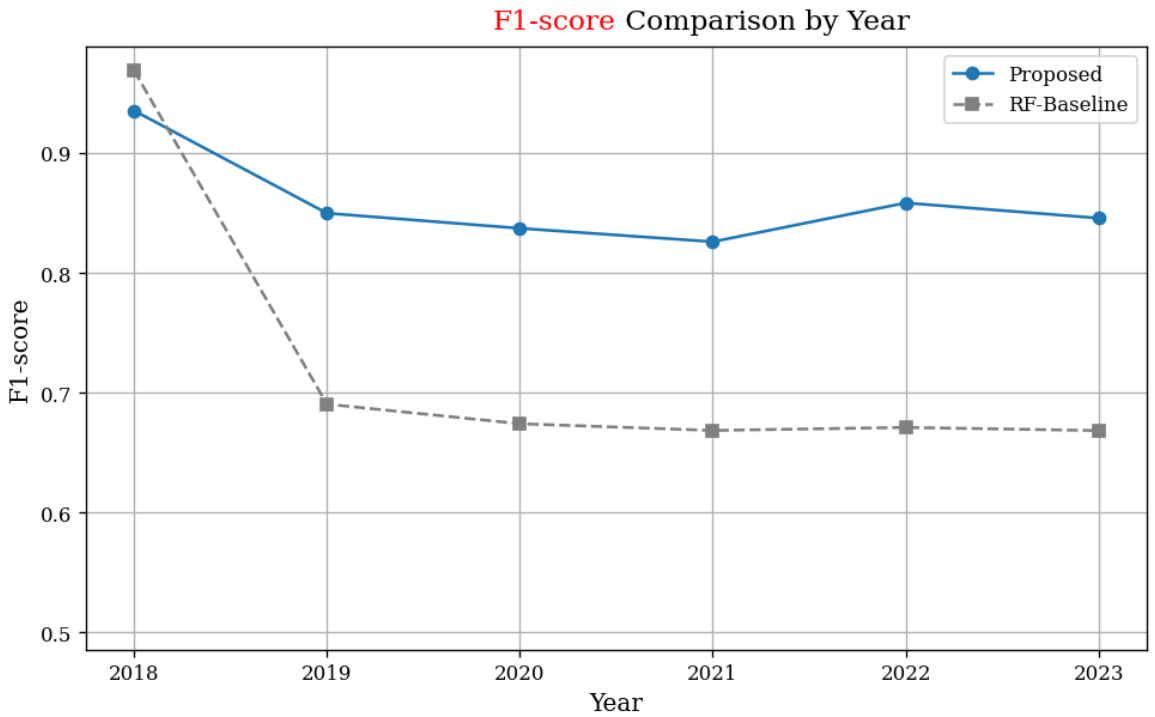
전체 성능 및 혼동 행렬

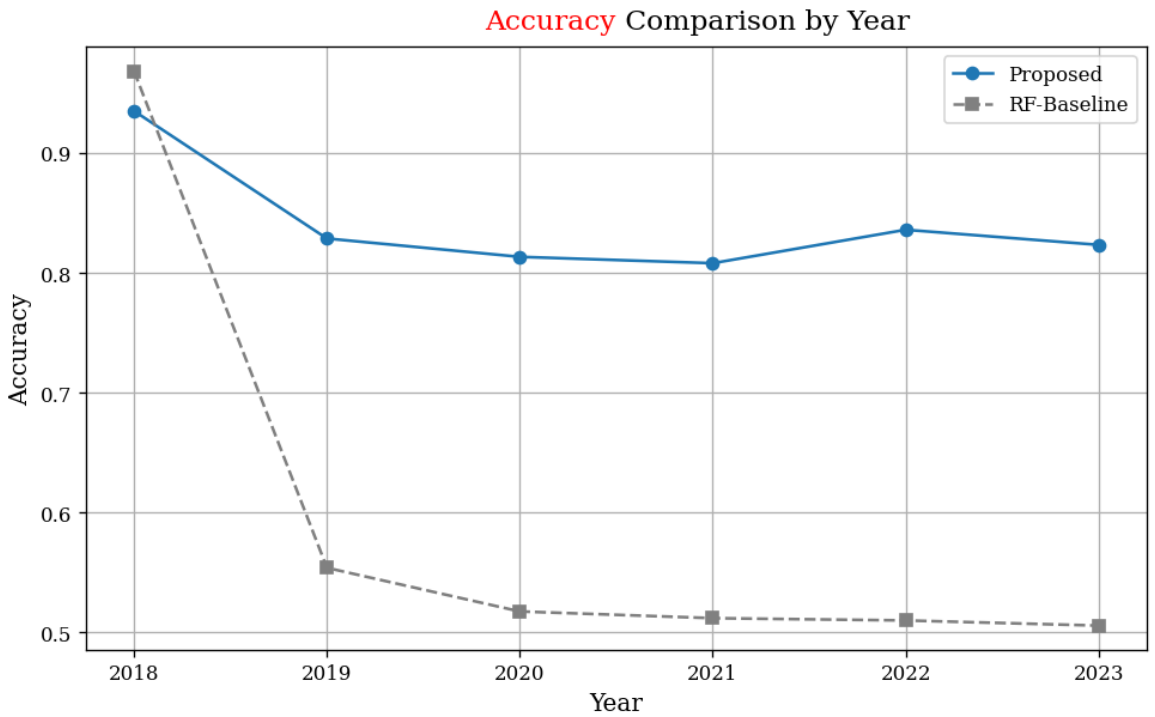
연도	정확도	F1-score
2018	0.9373	0.9378
2019	0.8243	0.8473
2020	0.8101	0.8352
2021	0.8096	0.8277
2022	0.8313	0.8553
2023	0.8169	0.8408
전체 평균	0.8384	0.8559

	예측 정상	예측 악성
실제 정상	12,813	5,034
실제 악성	749	17,181

결과 및 분석

Baseline과의 비교





04 결론 및 한계점

결론 및 한계점

한계

1. 커뮤니티 구조에 속하지 못하는 앱에 대해 고려 X
2. 커뮤니티 간 데이터 불균형 문제로 인해 일부 소형 커뮤니티에서 학습 제한 가능성 O
3. 정적 분석 데이터만을 사용 → 동적 행위 분석 불가능

기여점

1. API 호출을 동시 출현 그래프로 만들어 SNA의 일종인 Louvain 방법을 적용하여 커뮤니티 탐지
2. 커뮤니티 통계 정보를 통해 새로운 특징 정보를 만들어 반영
3. 재학습 없이도 비교적 높은 성능을 달성

Thank you

Q&A