ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 48, No. 1, pp. 35-42, 2021. 1
https://doi.org/10.5626/JOK.2021.48.1.35

# 텍스트 마이닝과 차원 축소 기법을 적용한 향상된 컨피규레이션 버그 리포트 예측

## (Improved Prediction for Configuration Bug Report Using Text Mining and Dimensionality Reduction)

최 정 환 † 최 지 원 †† 류 덕 산 ††† 김 순 태 ††††
(Jeongwhan Choi) (Jiwon Choi) (Duksan Ryu) (Suntae Kim)

**요 약** 소프트웨어 실패의 주요 원인들 중 하나로 컨피규레이션 버그가 있다. 소프트웨어 조직들은 이슈 트래킹 시스템을 통해 버그 리포트들을 수집하고 관리하는데, 버그 할당자는 해당 버그가 컨피규레이션 버그인지 식별하는데 시간을 소비할 수 있다. 컨피규레이션 버그를 예측하는 방법을 통해 버그 할당자의 의사 결정에 도움을 줘 노력을 줄일 수 있다. 본 논문에서는 텍스트 마이닝 기법과 차원 축소 기법을 이용하여 향상된 분류 모델을 제안한다. 본 논문은 6개의 오픈 소스 소프트웨어 프로젝트로부터 4,457개의 버그 리포트를 추출하고 컨피규레이션 버그 리포트를 분류하는 모델을 학습하고 예측 성능을 평가한다. 가장 좋은 성능을 보이는 방법은 Bag of Words로 피처를 추출하고 선형판별분석(LDA: Linear Discriminant Analysis)를 이용하여 피처의 차원을 축소 후 SMOTEENN 샘플링 기법을 이용하여 k-Nearest Neighbors 모델을 사용한다. 이에 대한 AUC 값은 0.9812이고 MCC가 0.942이다. 이는 Xia et al.의 방법보다 더 좋은 성능을 보이며, 이전 연구에서의 클래스 불균형 문제를 해결한다. 이러한 향상된 컨피규레이션 버그 리포트 예측을 통해, 이를 버그 할당자의 의사 결정에 필요한 정보를 줄 수 있거나 시간을 단축시킬 수 있다.

**키워드:** 컨피규레이션 버그 리포트, 선형판별분석, 차원축소, 클래스 불균형, 샘플링

***Abstract*** Configuration bugs are one of the main causes of software failure. Software organizations collect and manage bug reports using an issue tracking system. The bug assignor can spend excessive amounts of time identifying whether a bug is a configuration bug or not. Configuration bug prediction can help the bug assignor reduce classification efforts and aid decision making. In this paper, we propose an improved classification model using text mining and dimensionality reduction. This paper extracts 4,457 bug reports from six open-source software projects, trains a model to classify configuration bug reports, and evaluates prediction performance. The best performance method is obtained using the k-Nearest Neighbors model with the SMOTEENN sampling technique after extracting the feature with Bag of Words and then reducing the dimension of the feature using Linear Discriminant Analysis. The results show that ROC-AUC is 0.9812 and MCC is 0.942. This indicates better performance than Xia

· 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2019R1G1A1005047 & NRF-2020R1F1A1072039)

† 학생회원 : 연세대학교 인공지능학과 학생
jeongwhan.choi@yonsei.ac.kr
†† 학생회원 : 전북대학교 소프트웨어공학과 학생
wanny_94@jbnu.ac.kr
††† 종신회원 : 전북대학교 소프트웨어공학과 교수
(Jeonbuk Nat'l Univ.)
duksan.ryu@jbnu.ac.kr
(Corresponding author임)
†††† 정 회 원 : 전북대학교 소프트웨어공학과 교수
stkim@jbnu.ac.kr

논문접수 : 2020년 3월 19일
(Received 19 March 2020)
논문수정 : 2020년 11월 11일
(Revised 11 November 2020)
심사완료 : 2020년 11월 12일
(Accepted 12 November 2020)


정보과학회논문지 제48권 제1호(2021. 1)

et al.'s method and solves the class imbalance problem of our previous study. By predicting these enhanced configuration bug reports, our proposed approach can provide the bug assignors with information they need to make informed decisions.

**Keywords:** configuration bug report, linear discriminant analysis, dimensionality reduction, class imbalance, sampling

## 1. 서 론

Configuration 버그는 소프트웨어 실패의 주요한 원인들 중 하나이다[1]. 특히 오늘날의 대규모 소프트웨어 시스템들은 변화가 잦기 때문에 실패가 발생하고 이를 해결하기 위해 롤백이나 복구 메커니즘을 사용하는데, 이들은 Configuration 버그를 제어하는데 효과적이지 않다[2].

Oppenheimer et al.의 연구에 따르면 Configuration 버그가 대규모 시스템의 가동 중지 시간(downtown)의 주요 원인 중 하나라고 한다[2]. 또한, 2011 년도에 Facebook 은 Configuration 버그로 약 500 만명의 사용자들이 Facebook 웹 사이트에 접근하지 못하기도 하였다[3]. Yin et al.의 연구에 따르면 높은 심각성을 가진 케이스의 경우의 원인들 중 31%가 소프트웨어 시스템의 Configuration 버그와 관련이 있다고 말한다[4]. 따라서 Configuration 버그들은 미리 식별하지 않으면 계속해서 잠재적인 오류를 가지고 있을 수 있기 때문에 이러한 문제를 빠르게 식별하는 것이 중요하다[5].

이러한 버그들을 이슈 트래킹 시스템을 통해 관리하고, 이슈 리포트들을 분류하여 각각의 버그들을 개발자에게 할당한다. 할당하는 과정에서 Configuration 버그인지 인식하는데 막대한 시간을 소비할 수 있다. 해당 버그 리포트가 Configuration 버그인지 분류하는 자동화된 기법을 통해서 버그 할당하는 의사결정에 도움이 되고 시간 소비를 감소할 수 있기에 이 연구에 대한 동기가 된다.

본 논문에서는 자연어로부터 차원 축소 기법과 샘플링 기법, 분류 모델을 이용하여 Configuration 버그 리포트를 예측하는 향상된 기법을 제안한다. 6개의 오픈 소스 프로젝트의 버그 리포트들을 바탕으로 우리가 제안한 방법을 평가하고 비교한다. 그 결과 BoW와 LDA, SMOTEENN, KNN을 사용하는 모델이 평균 ROC-AUC와 MCC가 각각 0.9812, 0.942의 결과로 가장 좋은 성능을 보인다.

본 논문에서 기여하는 것들은 다음과 같다.

1) 자연어를 이용하여 Configuration 버그 리포트를 예측하는데 피쳐 선택 기법 대신 차원 축소 기법을 사용하는 방법을 제안한다.
2) 6개의 프로젝트, 4,457개의 버그 리포트들을 토대로 Xia et al.[6]의 방법보다 더 향상된 예측 성능을 가진다는 것을 보인다.

그리고 본 논문은 이전 연구에 대한 확장 논문이며 아래 항목을 추가하여 연구를 진행하였다[7].

1) 동일한 Java 언어 기반 프로젝트인 TOMEE데이터를 추가하여 531개의 버그리포트를 확보하였다.
2) 이전 연구에서 불균형한 클래스로 인한 문제가 유효성에 대한 위협이였지만, 본 논문에서는 오버샘플링과 언더샘플링 기법의 조합을 사용하여 이에 대한 위협을 해결하고자 하였다.
3) 분류 모델로 KNN과 Logistic Regression을 추가 실험하여 비교하였다.

## 2. 관련 연구

Configuration 버그에 대한 많은 연구가 있었다[6,8,9-12]. Attariyan과 Flinn은 데이터 흐름 분석을 활용하여 Configuration 버그를 추적하는 ConfAid를 개발하였다[8]. Zhang과 Ernst는 통계 분석을 활용하여 Configuration 오류를 식별하는 ConfDiagnoser라는 정적 분석과 동적 프로파일링을 결합해 Configuration 버그의 루트를 식별하고자 하는 노력을 기울였다[9]. Arshad et al.은 GlassFish와 JBoss로부터 Configuration 버그를 추출하고 해당 버그의 특징을 problem-type, problem-time과 같은 관점에서 나타내었다[11].

Xia et al.[6]은 Configuration 버그 리포트를 키워드 기반으로 분류하는 방법을 제안한다. 이 연구에서는 BoW로 피쳐를 추출하고 Chi-square 또는 Information Gain 기법들을 사용한다. 이를 통해 상위 10%의 피쳐들을 선택하여 사용한다. 그리고 Naïve Bayes Multinomial 분류 모델을 사용하는 것이 가장 좋은 예측 성능을 보인다.

Bowen et al.의 연구에서는 단일 피쳐 선택 기법을 사용하는 대신 5가지 기법들을 계산하여 조합한 후 피쳐들을 선택하는 방법을 제안한다[13].

CoLUA를 제안한 Wen et al.[12]은 Mozilla, Apache, MySQL 3가지의 프로젝트를 사용하여 Configuration 버그를 예측한다. 이 연구는 Xia et al.의 방법과 유사하게 Chi-square와 Information Gain을 계산 후 피쳐들을 선택한다. 그리고 Naïve Bayes, Logistic Regression, Decision Tree들을 사용하여 예측한다.

## 3. 접근 방법

### 3.1 전체 프레임워크

그림 1은 본 논문의 전체적인 접근 방법을 나타낸다. 먼저 단계 1에서는 Apache JIRA 이슈 트래킹 시스템으로부터 버그 리포트들을 마이닝하고 데이터셋을 구축한다. 총 6개의 오픈 소스 프로젝트(ACCUMUIO, ACTIVEMQ, CAMEL, FLUME, TOMEE, WICKET)들을 사용한다[14-19]. 이 프로젝트들의 버그 리포트 개수는 총 4,457 개이다. 버그 리포트는 자연어 중 요약문과 설명문을 포함하고 있는데 요약문만을 사용한다.

단계 2에서 텍스트에 대한 전처리 과정을 가진다. 텍스트들을 토큰화하고 불용어들을 제거한다. 단어들을 소문자로 모두 변환하고 특수문자들을 제거하고 어근화한다.

단계 3에서는 BoW(Bag of Words)으로 토큰화된 텍스트 데이터들을 표현한다. 실험 목차에서는 피쳐를 BoW 뿐만 아니라 워드 임베딩 기법(Word2Vec, FastText, GloVe)들을 사용해서 비교한다.

단계 4에서는 이전 단계에서 추출한 피쳐들을 차원 축소 기법(PCA, LDA)을 사용하는 과정이다.

단계 5에서는 본 논문의 데이터셋의 클래스가 불균형하기 때문에 이를 해결하기 위해 오버샘플링과 언더샘플링 기법의 조합인 SMOTEENN 기법을 사용한다. 이를 통해 불균형한 클래스의 크기를 해결하고자 한다.

단계 6에서는 차원 축소를 진행한 피쳐들을 KNN, SVM, MLP, Logistic Regression 모델로 학습한 후 각 분류 모델의 성능을 평가한다. KNN의 경우 neighbor 수를 3으로 MLP는 뉴런 크기가 512인 은닉층 두 개를 사용한다.

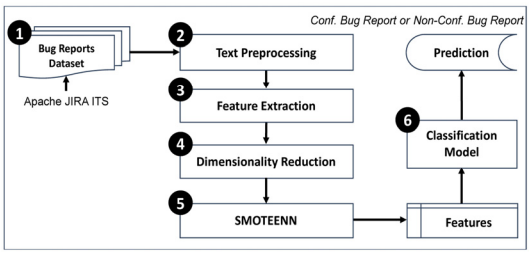


그림 1 전체 프레임 워크
Fig. 1 Overall Framework

### 3.2 차원 축소 기법

본 연구에서는 차원 축소 기법으로 LDA 또는 PCA를 사용한다. LDA(Linear Discriminant Analysis)는 특징 공간상에서 클래스 분리를 최대화하는 주축으로 사상시켜 선형 서브공간으로 차원을 축소한다. 본 연구에서의 데이터셋의 클래스가 두 개이기 때문에, 하나의 피쳐로 축소되어 사용한다[20]. PCA(Principal Component Analysis)는 상관 관계가 없는 새로운 변수를 찾고 최대한 많은 변동성을 유지하기 위해 분산을 최대화한다[21]. PCA를 사용하는 경우 200 개의 피쳐로 축소하여 사용한다.

### 3.3 오버샘플링과 언더샘플링 기법

본 연구에서는 불균형한 클래스 문제를 다루기 위해 오버샘플링 기법과 언더샘플링 기법을 조합하여 사용한다. SMOTEENN은 SMOTE(Synthetic Minority Oversampling Technique)와 ENN(Edited Nearest Neighbours)의 조합이다. 먼저 SMOTE를 사용하여 오버샘플링을 진행한 후 ENN을 이용해 샘플들을 제거한다[22,23]. SMOTETomek은 SMOTE와 언더샘플링 기법인 Tomek 링크의 조합이다[24,25].

## 4. 실험 환경 설정

### 4.1 연구 질문

본 연구에서는 4 가지의 연구 질문을 설정한다. 각 연구 질문에 대한 통계적 유의함을 분석하기 위해 효과 크기를 사용한다. Sawilowsky가 구분한 Cohen's D 값에 따른 효과 크기를 참고하여 분석한다[26].

**RQ1** 본 연구에서 제안하는 방법이 관련 연구인 Xia et al.의 방법보다 얼마나 더 좋은 성능을 보이는가?

$H_0$ 본 논문에서 제안하는 모델이 Xia et al.에서 제안하는 모델보다 성능이 같거나 낮다.

$H_A$ 본 논문에서 제안하는 모델이 Xia et al.에서 제안하는 모델보다 성능이 높다.

본 논문에서는 BoW로 피쳐들을 추출한 후 차원 축소 기법인 LDA와 SMOTEENN을 사용하는 방법을 제안한다. 분류 모델로는 MLP, SVM, KNN, Logistic Regression을 사용한다. 이 연구 질문에 답하기 위해서 Chi-square의 피쳐 선택 기법을 적용하고 Naïve Bayes Multinomial 모델을 제안한 Xia et al.의 방법과 비교한다[6,27].

**RQ2** 차원 축소의 효과는 얼마나 영향을 미치는가?

$H_0$ 차원 축소 기법을 사용하는 방법이 이를 사용하지 않은 방법보다 성능이 같거나 낮다.

$H_A$ 차원 축소 기법을 사용하는 방법이 이를 사용하지 않은 방법보다 성능이 높다.

본 연구에서는 차원 축소 기법으로 LDA를 사용하며, 이를 사용하는 방법과 사용하지 않은 방법을 비교한다.

**RQ3** 차원 축소의 효과는 워드 임베딩만을 사용할 때와 비교할 때 어느 정도 효과가 있는가?

$H_0$ 차원 축소 기법이 워드 임베딩 기법보다 성능이 같거나 낮다.

$H_A$ 차원 축소 기법이 워드 임베딩 기법보다 성능이 높다.

BoW는 상당히 많은 피쳐 수가 생성된다. 반면에 워드 임베딩 방법을 사용하면 피쳐 수를 조정할 수 있다. 본 연구에서 제안하는 방법은 BoW으로 피쳐를 추출 후, 차원 축소를 통해 피쳐의 수를 감소시킨다. 이 연구 질문에 답하기 위해 BoW으로 표현된 피쳐들을 차원 축소한 결과가 워드 임베딩(Word2Vec, GloVe, FastText)보다 좋은 효과가 있음을 가설을 통해 분석한다.

**RQ4** 불균형한 클래스를 해결하기 위해 사용한 기법은 성능에 얼마나 영향을 미치는가?

$H_0$ SMOTEENN 기법과 SMOTETomek을 사용한 기법이 이를 사용하지 않은 기법보다 성능이 같거나 낮다.

$H_A$ SMOTEENN 기법과 SMOTETomek을 사용한 기법이 이를 사용하지 않은 기법보다 성능이 높다.

불균형한 클래스를 해결하기 위해SMOTE기법으로 오버샘플링 후 ENN 또는 Tomek 링크로 언더샘플링을 한다. 이의 효과를 알기 위해 샘플링 기법을 사용한 방법과 이를 사용하지 않은 방법을 비교한다.

### 4.2 실험 환경

실험은 3.1GHz CPU, 8GB RAM의 MacOS에서 Python을 이용하였다. 데이터셋은 Jira 이슈 시스템에서 수집한 버그리포트들과 커밋 정보들로부터 구축한다. 그리고 Configuration 버그 리포트를 예측하기 위한 데이터셋을 만들기 위해서는 라벨링 작업을 해야 하는데, 해당 버그인지 아닌지 매뉴얼리하게 분석하여 라벨링하였다. 분석을 위해서 해당 버그리포트의 커밋 정보를 참고하였다. 라벨링 한 결과 각각의 버그 리포트의 개수는 표 1과 같다.

본 논문에서의 데이터 클래스는 불균형한 분포가 나타나는데 이를 다루기 위해 Stratified 10-fold 교차검증으로 각 fold를 나눈 후 SMOTEENN을 적용하여 실험한다.

### 4.3 평가 척도

4.3.1 F-measure

F-measure는 정확률(precision)과 재현률(recall)두 값의 조화 평균이며 정확률의 증가가 재현률보다 큰지 그렇지 않은지를 평가한다[28]. 이전의 많은 소프트웨어공학 논문에서 정확률과 재현률 사이의 트레이드 오프 때문에 F-measure가 주요 평가 척도로 사용되었다. 아래와 같이 Configuration의 F-measure는 F(C), Non-Configuration F-measure는 F(NC)로 표현된다.

$$F(C) = \frac{2 \cdot Precision(C) \cdot Recall(C)}{Precision(C) + Recall(C)} \quad (1)$$

$$F(NC) = \frac{2 \cdot Precision(NC) \cdot Recall(NC)}{Precision(NC) + Recall(NC)} \quad (2)$$

표 1 수집된 데이터셋의 통계
Table 1 Statistics of Collected Data

| Project | Time | #Bug Reports | #Config. Bug Reports |
|---|---|---|---|
| ACCUMULO | 2011.10~2017.09 | 532 | 55 |
| ACTIVEMQ | 2009.01~2018.10 | 913 | 81 |
| CAMEL | 2009.01~2018.10 | 1249 | 158 |
| FLUME | 2010.06~2017.07 | 336 | 40 |
| TOMEE | 2011.06~2018.12 | 531 | 61 |
| WICKET | 2009.01~2018.07 | 896 | 14 |
| Total | | 4,457 | 409 |

4.3.2 ROC-AUC

ROC-AUC(Receiver Operating Characteristic-Area Under the Curve)는 ROC 커브 그래프의 면적 값을 이용한다[29]. 이 그래프는 모든 분류 임계값에서 분류 모델의 성능을 나타내는데 TPR(True Positive Rate)와 FPR(False Positive Rate)의 변화를 시각화 한 것이다.

$$TPR = \frac{TP}{TP + FN} \quad (3) \qquad FPR = \frac{FP}{FP + TN} \quad (4)$$

4.3.3 MCC

MCC(Matthews Correlation Coefficient)는 불균형한 클래스에 대한 분류 문제에 적합한 평가 척도이다[30]. MCC는 혼동 행렬 범주 모두에서 좋은 결과를 얻은 경우에만 높은 값을 가지는 안정적인 통계 비율이다. 이는 식 (5)와 같이 정의하며 -1에서 +1 사이의 값을 갖는다.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}} \quad (5)$$

## 5. 실험 결과

### 5.1 RQ1: 본 논문의 방법 vs. Xia et al.의 방법

Xia et al.에서는 각 피쳐들의 Chi-square를 계산하여 상위 10%의 피쳐들을 선택한다. 이들이 제안한 방법을 이용했을 때, 각각의 프로젝트에 대한 평균 결과는 표 2와 같이 F(C)와 F(NC)는 각각 0.6212, 0.9528이다. AUC값은 0.8612, MCC 값은 0.6097의 값을 보인다. 이는 각각의 Configuration 버그 리포트의 예측력이 낮은 것을 알 수 있다. 그리고 불균형한 클래스로 인해 Configuration 버그 리포트가 아닌 것은 예측이 잘 되는 것을 볼 수 있다.

반면에 본 연구에서의 방법은 표 3에서와 같이 LDA와 KNN을 사용할 때가 가장 좋은 성능을 보인다. PCA와 KNN을 사용한 경우에는 AUC가 0.6548, MCC가 0.3417로 LDA 차원 축소 기법보다 현저히 낮았다.

실험 결과를 통해서 귀무 가설을 기각할 수 있는지 분석한다. 이를 위해 본 논문에서 제안하는 방법(BoW+LDA+SMOTEENN+KNN)과 관련연구에서 가장 좋은

표 2 각 프로젝트에 대한 관련연구의 방법에 대한 실험 결과
Table 2 Experimental Results of the Method of Related Work for Each Project

| Xia et al.'s Approach: BoW+Chi-square+NBM | | | | |
|---|---|---|---|---|
| Project | F(C) | F(NC) | AUC | MCC |
| ACCUMULO | 0.661 | 0.954 | 0.8518 | 0.6421 |
| AMQ | 0.63 | 0.959 | 0.8306 | 0.5958 |
| CAMEL | 0.694 | 0.956 | 0.825 | 0.6493 |
| FLUME | 0.626 | 0.923 | 0.884 | 0.6015 |
| TOMEE | 0.606 | 0.939 | 0.8143 | 0.5619 |
| WICKET | 0.51 | 0.986 | 0.9614 | 0.6078 |
| Average | 0.621 | 0.953 | 0.861 | 0.610 |

방법인 BoW+Chi-square+NBM 모델을 비교한다. 그리고 각 모델의 MCC에 대한 성능 차이를 효과 크기를 이용해 분석한다. 효과 크기인 Cohen's D의 값은 2.11이다. 이는 Cohen에 따르면 large에 해당하며, Sawilowsky에 따르면 D 값이 2.0보다 크기 때문에 영향이 크다는 것을 알 수 있다[26]. 이를 통해 두 모델의 성능 차이가 통계적으로 유의함을 보임을 알 수 있다. 따라서 대립 가설인 "본 논문에서 제안하는 모델이 Xia et al. 에서 제안하는 모델보다 성능이 높다"를 채택할 수 있다.

### 5.2 RQ2: 차원 축소 LDA의 효과

차원 축소 기법인 LDA 기법의 효과를 분석하기 위해 본 연구에서 제안하는 방법과 차원 축소를 사용하지 않은 BoW+SMOTEENN+KNN 방법을 비교한다. 차원 축소 기법인 LDA를 사용하지 않았을 때, 각 프로젝트에 대한 평균 AUC와 MCC는 표 4에서 알 수 있듯이 0.5012와 0.0061이다. 본 연구에서 제안한 방법인 Bow+LDA+SMOTEENN+KNN 모델과 Bow+SMOTEENN+KNN 모델을 비교해 효과 크기를 분석한 결과 Cohen's D의 값이 4.503이다. 이는 효과가 매우 크고, 본 연구에서 세운 대립 가설을 채택할 수 있다.

### 5.3 RQ3: BoW+LDA vs. 워드 임베딩

표 5는 Word2Vec, FastText, GloVe의 워드 임베딩을 사용한 각각의 프로젝트에 대한 실험 결과를 나타낸다. 각각에 대한 AUC와 MCC 평균 값은 GloVe+LDA+SMOTEENN+MLP 방법이 0.6274, 0.1753으로 워드 임베딩 방법을 사용한 것 중에서 가장 높은 값을 보여준다. 반면에 Word2Vec는 각각 0.5498, 0.0579, FastText는 각각 0.5961, 0.1059으로 전반적으로 낮은 성능을 보인다.

표 3 차원 축소 기법으로 LDA 사용한 본 연구의 접근 방법의 실험 결과
Table 3 Results of our Approach using LDA as a Dimension Reduction Technique

| Our Approach: | Metrics | Project | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | ACCUMULO | AMQ | CAMEL | FLUME | TOMEE | WICKET | Average |
| BoW+LDA+SMOTEENN+KNN | F(C) | 0.982 | 0.982 | 0.991 | 0.864 | 0.896 | 0.897 | 0.9353 |
| | F(NC) | 0.998 | 0.998 | 0.999 | 0.979 | 0.985 | 0.998 | 0.9928 |
| | AUC | 0.9906 | 0.9906 | 0.996 | 0.9583 | 0.9778 | 0.9739 | 0.9812 |
| | MCC | 0.9807 | 0.9806 | 0.9893 | 0.877 | 0.8915 | 0.9327 | 0.942 |
| BoW+LDA+SMOTEENN+MLP | F(C) | 0.931 | 0.913 | 0.969 | 0.857 | 0.857 | 0.8 | 0.8878 |
| | F(NC) | 0.992 | 0.991 | 0.995 | 0.978 | 0.978 | 0.996 | 0.9878 |
| | AUC | 0.9844 | 0.9797 | 0.9954 | 0.9674 | 0.9715 | 0.996 | 0.9824 |
| | MCC | 0.926 | 0.9132 | 0.966 | 0.8675 | 0.8568 | 0.8705 | 0.9 |
| BoW+LDA+SMOTEENN+SVM | F(C) | 0.973 | 0.963 | 0.981 | 0.857 | 0.896 | 0.875 | 0.9242 |
| | F(NC) | 0.997 | 0.996 | 0.997 | 0.978 | 0.985 | 0.998 | 0.9918 |
| | AUC | 0.9896 | 0.9851 | 0.9946 | 0.9674 | 0.9778 | 0.9977 | 0.9854 |
| | MCC | 0.971 | 0.9604 | 0.9793 | 0.8675 | 0.8926 | 0.9194 | 0.9317 |
| BoW+LDA+SMOTEENN+Logistic Regression | F(C) | 0.982 | 0.975 | 0.994 | 0.868 | 0.898 | 0.889 | 0.9343 |
| | F(NC) | 0.998 | 0.998 | 0.999 | 0.983 | 0.987 | 0.998 | 0.9938 |
| | AUC | 0.9906 | 0.9863 | 0.9964 | 0.9075 | 0.9291 | 0.9494 | 0.9599 |
| | MCC | 0.9807 | 0.973 | 0.993 | 0.8537 | 0.8846 | 0.8812 | 0.9277 |

표 4 BoW+SMOTENN+KNN 방법의 성능 평가 결과
Table 4 Performance Evaluation Results of the BoW+SMOTENN+KNN

| BoW+ | Metrics | Project | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | ACCUMULO | AMQ | CAMEL | FLUME | TOMEE | WICKET | Average |
| SMOTEENN+KNN | F(C) | 0.187 | 0.213 | 0.223 | 0.213 | 0.207 | 0.031 | 0.179 |
| | F(NC) | 0.029 | 0.007 | 0.016 | 0.007 | 0.013 | 0.014 | 0.0143 |
| | AUC | 0.499 | 0.5017 | 0.4979 | 0.5017 | 0.5032 | 0.5034 | 0.5012 |
| | MCC | 0.006 | 0.007 | 0 | 0.006 | 0.012 | 0.0054 | 0.0061 |

표 5 워드 임베딩 기법을 이용한 방법의 실험 결과
Table 5 Experimental Results of Method using Word Embedding Techniques

| | Metrics | Project | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | ACCUMULO | AMQ | CAMEL | FLUME | TOMEE | WICKET | Average |
| Word2Vec+ SMOTEENN+ KNN | F(C) | 0.218 | 0.163 | 0.218 | 0.239 | 0.195 | 0.049 | 0.1803 |
| | F(NC) | 0.32 | 0.421 | 0.277 | 0.278 | 0.494 | 0.647 | 0.4062 |
| | AUC | 0.587 | 0.5113 | 0.4946 | 0.5696 | 0.4963 | 0.64 | 0.5498 |
| | MCC | 0.1378 | 0.0139 | 0 | 0.1207 | 0 | 0.0747 | 0.0579 |
| GloVe+LDA+ SMOTEENN+ MLP | F(C) | 0.292 | 0.235 | 0.312 | 0.301 | 0.297 | 0.093 | 0.255 |
| | F(NC) | 0.737 | 0.845 | 0.704 | 0.758 | 0.768 | 0.978 | 0.7983 |
| | AUC | 0.6765 | 0.608 | 0.6527 | 0.6443 | 0.648 | 0.5347 | 0.6274 |
| | MCC | 0.2222 | 0.1416 | 0.206 | 0.1952 | 0.2038 | 0.0827 | 0.1753 |
| FastText+ SMOTEENN+ KNN | F(C) | 0.196 | 0.183 | 0.299 | 0.276 | 0.213 | 0.05 | 0.2028 |
| | F(NC) | 0.284 | 0.516 | 0.617 | 0.56 | 0.499 | 0.654 | 0.5217 |
| | AUC | 0.5304 | 0.5612 | 0.6459 | 0.6349 | 0.5354 | 0.6688 | 0.5961 |
| | MCC | 0.0521 | 0.0743 | 0.1972 | 0.1817 | 0.0467 | 0.0836 | 0.1059 |

표 6 SMOTETomek을 사용한 방법과 샘플링 기법을 사용하지 않은 방법에 대한 실험 결과
Table 6 Experimental Results for Method using SMOTETomek and the Method without Sampling Technique

| | Metrics | Project | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | ACCUMULO | AMQ | CAMEL | FLUME | TOMEE | WICKET | Average |
| BoW+LDA+ SMOTETomek+ KNN | F(C) | 0.218 | 0.163 | 0.218 | 0.239 | 0.195 | 0.049 | 0.1803 |
| | F(NC) | 0.32 | 0.421 | 0.277 | 0.278 | 0.494 | 0.647 | 0.4062 |
| | AUC | 0.587 | 0.5113 | 0.4946 | 0.5696 | 0.4963 | 0.64 | 0.5498 |
| | MCC | 0.1378 | 0.0139 | 0 | 0.1207 | 0 | 0.0747 | 0.0579 |
| BoW+LDA+ KNN | F(C) | 0.292 | 0.235 | 0.312 | 0.301 | 0.297 | 0.093 | 0.255 |
| | F(NC) | 0.737 | 0.845 | 0.704 | 0.758 | 0.768 | 0.978 | 0.7983 |
| | AUC | 0.6765 | 0.608 | 0.6527 | 0.6443 | 0.648 | 0.5347 | 0.6274 |
| | MCC | 0.2222 | 0.1416 | 0.206 | 0.1952 | 0.2038 | 0.0827 | 0.1753 |

본 연구에서 제안한 방법과 GloVe+LDA+SMOTEENN+MLP모델을 비교해 효과 크기를 분석한 결과 Cohen's D의 값이 8.538이다. 이는 Sawilowsky에 따르면 효과가 매우 크고, 본 연구의 대립 가설인 "차원 축소 기법을 사용하는 방법이 이를 사용하지 않은 방법보다 성능이 높다"를 채택할 수 있다.

### 5.4 RQ4: SMOTEENN의 효과

오버샘플링과 언더샘플링의 조합인 SMOTEENN과 SMOTETomek을 사용하였을 때, SMOTEENN의 AUC와 MCC 값이 0.9812, 0.942의 결과가 나온다. SMOTETomek의 AUC와 MCC 값은 0.9781, 0.8966으로 본 연구에서는 불균형한 클래스를 해결하기 위해 SMOTETomek 보다 성능이 우수한 SMOTEENN을 사용한다.

SMOTEENN의 효과를 알기 위해 BoW+LDA+SMOTEENN+KNN과 BoW+LDA+KNN의 결과를 비교한다. 샘플링 기법을 사용하지 않은 경우에는 AUC 값이 0.9566, MCC값이 0.9086의 결과가 나온다. 이는 표 6에서 알 수 있듯이 샘플링 기법을 사용하지 않을 경우에 AUC가 감소하는 것을 알 수 있다. 반면에, F(NC)는 변화가 적지만 F(C)에서는 SMOTEENN을 사용할 때 보다 작은 것을 알 수 있다. 하지만 SMOTETomek은 샘플링 기법을 사용하지 않은 방법보다 F(C)와 MCC가 낮은 것을 알 수 있다. 따라서 본 연구에서 제안하는 방법인 BoW+LDA+SMOTEENN+KNN 모델과 샘플링 기법을 사용하지 않은 방법인 BoW+LDA+KNN 방법을 비교한다. 효과 크기를 분석한 결과 Cohen's D의 값이 0.19로, 효과 크기가 작아 대립가설을 채택할 수 없다. 귀무가설을 기각하지 못하기에 클래스 불균형 학습 기법이 configuration 버그 리포트 예측 성능에 작은 영향만을 준다는 것을 확인할 수 있다.

## 6. 위협 요소

본 논문에서 데이터셋 구축을 위해 해당 버그 리포트가 Configuration 버그 리포트인지 아닌지 직접 라벨링을 하였는데, 이 부분이 내부적인 유효성에 대한 위협이다. 이러한 위협을 완화시키기 위해 라벨링 후 데이터셋에 대한 검토 과정을 가졌다. 외부 유효성에 대한 위협은 일반화와 관련이 있는데, 6개의 프로젝트로부터 4,457개의 버그 리포트들을 바탕으로 실험을 하였다. Xia et al.의 연구에서는 5개의 프로젝트로부터 3,203 개의 버그

리포트를 사용하였다[6]. 본 연구에서는 TOMEE 프로젝트를 추가함으로써 외부 유효성에 대한 위협을 줄이려고 노력했다. 구성 유효에 대한 위협은 본 논문에서 사용한 평가 척도와 관련이 있다. 평가 척도로 F-measure 이외에 AUC와 MCC를 사용하여, 이 위협을 완화하였다.

## 7. 결 론

본 논문에서, 버그 리포트의 자연어로부터 피쳐들을 추출하고 차원 축소 기법과 불균형 클래스를 다루는 기법, 분류 모델을 이용하여 Configuration 버그 리포트를 예측하는 방법을 제안한다. 네 가지 피쳐 추출(BoW, Word2Vec, FastText, GloVe) 방법과 두 가지 차원 축소 기법(LDA, PCA)와 샘플링 기법(SMOTEENN, SMOTETomek), 네 가지 분류 모델(MLP, SVM, KNN, Logistic Regression)들을 조합하여 실험하였다. 본 논문은 6개의 오픈 소스 프로젝트의 총 4,457 개의 버그 리포트들을 바탕으로 우리가 제안한 방법을 평가하였다. 실험 결과로는 KNN 모델과 LDA 차원 축소 기법, SMOTEENN을 사용한 결과가 예측 성능이 가장 높았다. 6개의 프로젝트에 대한 평균 값으로 각각의 F(C)와 F(NC), AUC, MCC는 0.9353, 0.9928, 0.9812, 0.942이다. 본 연구 방법이 관련 연구와 이전 연구보다 향상되었다는 것을 확인하기 위해 통계적 검정을 사용한다.

향상된 정확한 Configuration 버그 리포트 예측을 통해, Configuration 버그라는 것을 인식하는 리소스를 감소할 수 있다. 이로 인해 버그 수정 프로세스에서의 버그 할당하는 부분에서 도움을 줄 수 있는 기대효과가 있다.

향후 연구로는 버그 리포트에서 자연어 뿐만 아니라 다른 요소들인 컴포넌트, 우선순위, 스택트레이스 정보 등에 대해서도 고려할 계획이며, 분류 모델로써 Neural Network 기반 모델들도 실험할 계획이다.

## References


[1] B. MAURER, "Fail at Scale: Reliability in the Face of Rapid Change," *Commun. oft he ACM58*, No. 11, pp. 44-49, 2015.

[2] D. Oppenheimer, D. a. Patterson, and A. Ganapathi, "Why do Internet Services fail, and what can be done about it?," *Proc. 4th Conf. USENIX Symp. Internet Technol. Syst.*, 4, 2003.

[3] "More Details on Today's Outage | Facebook." [Online]. Available: https://www.facebook.com/notes/facebook-engineering/more-details-on-todays-outage/431441338919/. [Accessed: 03-Jan-2020].

[4] Z. Yin, X. Ma, J. Zheng, Y. Zhou, L. N. Bairavasundaram, and S. Pasupathy, "An empirical study on configuration errors in commercial and open source systems," *SOSP'11 - Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, 2011.

[5] T. Xu et al., "Early Detection of Configuration Errors to Reduce Failure Damage This paper is included in the Proceedings of the," *Osdi*, pp. 619-634, 2016.

[6] X. Xia, D. Lo, W. Qiu, X. Wang, and B. Zhou, "Automated configuration bug report prediction using text mining," *Proc. - Int. Comput. Softw. Appl. Conf.*, pp. 107-116, 2014.

[7] J. Choi, J. Choi, D. Ryu, and S. Kim, "Prediction for Configuration Bug Report Using Text Mining," *Proc. of the 22nd Korea Conference on Software Engineering (KCSE 2020)*, pp. 350-357, 2020.

[8] M. Attariyan and J. Flinn, "Automating configuration troubleshooting with dynamic information flow analysis," *Design*, 2010.

[9] S. Zhang and M. D. Ernst, "Automated diagnosis of software configuration errors," *Proc. - International Conference on Software Engineering*, 2013.

[10] T. Xu et al., "Do not blame users for misconfigurations," *SOSP 2013 - Proceedings of the 24th ACM Symposium on Operating Systems Principles*, 2013.

[11] F. A. Arshad, R. J. Krause, and S. Bagchi, "Characterizing configuration problems in Java EE application servers: An empirical study with GlassFish and JBoss," *2013 IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013*, 2013.

[12] W. Wen, T. Yu, and J. H. Hayes, "CoLUA: Automatically Predicting Configuration Bug Reports and Extracting Configuration Options," *Proc. - International Symposium on Software Reliability Engineering, ISSRE,* 2016.

[13] B. Xu, D. Lo, X. Xia, A. Sureka, and S. Li, "EFSPredictor: Predicting configuration bugs with ensemble feature selection," *Proc. - Asia-Pacific Software Engineering Conference, APSEC, 2016*, Vol. 2016-May, pp. 206-213.

[14] "Accumulo - ASF JIRA." [Online]. Available: https://issues.apache.org/jira/projects/ACCUMULO/summary. [Accessed: 15-Mar-2020].

[15] "ActiveMQ - ASF JIRA." [Online]. Available: https://issues.apache.org/jira/projects/AMQ/summary. [Accessed: 15-Mar-2020].

[16] "Camel - ASF JIRA." [Online]. Available: https://issues.apache.org/jira/projects/CAMEL/summary. [Accessed: 15-Mar-2020].

[17] "Flume - ASF JIRA." [Online]. Available: https://issues.apache.org/jira/projects/FLUME/summary. [Accessed: 15-Mar-2020].

[18] "TomEE - ASF JIRA." [Online]. Available: https://issues.apache.org/jira/projects/TOMEE/summary. [Accessed: 15-Mar-2020].

[19] "Wicket - ASF JIRA." [Online]. Available: https://

issues.apache.org/jira/projects/WICKET/summary. [Accessed: 15-Mar-2020].
[20] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI Commun.*, 2017.
[21] "Principal component analysis and redundancy analysis," *Analysing Ecological Data*, 2007.
[22] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard, "Balancing Training Data for Automated Annotation of Keywords: a Case Study," *Proc. Second Brazilian Work. Bioinforma*, 2003.
[23] D. L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Trans. Syst. Man Cybern.*, 1972.
[24] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explor. Newsl.*, 2004.
[25] N. Thai-Nghe, "Learning optimal threshold on resampling data to deal with class imbalance," *Proc. IEEE RIVF …*, 2010.
[26] S. S. Sawilowsky, "New Effect Size Rules of Thumb," *J. Mod. Appl. Stat. Methods*, Vol. 8, No. 2, pp. 597-599, Nov. 2009.
[27] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys*. 2002.
[28] J. Han and M. Kamber, *Data Mining Concept and Tehniques*, 2006.
[29] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," *ACM International Conference Proceeding Series*, 2006.
[30] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric," *PLoS One*, 2017.

최 정 환

2020년 전북대학교 소프트웨어공학과 학사. 2020년~현재 연세대학교 인공지능학과 석박통합과정. 관심분야는 인공지능, 소프트웨어 결함예측, 그래프컨볼루션네트워크

최 지 원

2020년 전북대학교 소프트웨어공학과 학사. 2020년~현재 전북대학교 소프트웨어공학과 석사과정. 관심분야는 소프트웨어공학, 인공지능

류 덕 산

1999년 한양대학교 전자계산학과 학사
2012년 한국과학기술원 및 카네기멜론대학교 소프트웨어공학 복수학위 석사
2016년 한국과학기술원 전산학부 박사
2018년 9월~현재 전북대학교 소프트웨어공학과 조교수. 관심분야는 인공지능기반 소프트웨어분석, 소프트웨어 결함예측, 소프트웨어 신뢰성, 소프트웨어 메트릭스, 소프트웨어 품질보증, 자율시스템

김 순 태

2003년 중앙대학교 컴퓨터공학과 학사
2007년 서강대학교 컴퓨터공학과 석사
2010년 서강대학교 컴퓨터공학과 박사
2014년~현재 전북대학교 소프트웨어공학과 부교수. 관심분야는 소프트웨어공학, 블록체인/스마트컨트랙트, 인공지능