JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 11, pp. 131-138, November 2022
https://doi.org/10.9708/jksci.2022.27.11.131

Implementation of Git's Commit Message Complex Classification Model for Software Maintenance

Ji-Hoon Choi*, Joon-Yong Kim**, Seong-Hyun Park*

*Ph. Student, Dept. of Computer Engineering, Kongju National University, Chungnam, Korea
**Professor, Dept. of IT Convergence Software, Seoul Theological University, Gyeonggi-do, Korea
*Ph. Student, Dept. of Computer Engineering, Kongju National University, Chungnam, Korea

[Abstract]

Git's commit message is closely related to the project life cycle, and by this characteristic, it can greatly contribute to cost reduction and improvement of work efficiency by identifying risk factors and project status of project operation activities. Among these related fields, there are many studies that classify commit messages as types of software maintenance, and the maximum accuracy among the studies is 87%. In this paper, the purpose of using a solution using the commit classification model is to design and implement a complex classification model that combines several models to increase the accuracy of the previously published models and increase the reliability of the model. In this paper, a dataset was constructed by extracting automated labeling and source changes and trained using the DistillBERT model. As a result of verification, reliability was secured by obtaining an F1 score of 95%, which is 8% higher than the maximum of 87% reported in previous studies. Using the results of this study, it is expected that the reliability of the model will be increased and it will be possible to apply it to solutions such as software and project management.

▸ **Key words**: Commit Message, Multi-Label Classification, DistilBERT(Bidirectional Encoder Representations from Transformers), Source Change, Auto Labeling

[요 약]

Git의 커밋 메시지는 프로젝트 생명주기와 밀접한 연관성을 지니고 있으며, 이러한 특성에 의해 프로젝트 운영 활동의 위험요소와 프로젝트 현황 등을 파악하여 비용 절감과 작업효율 개선 등에 큰 기여를 할 수 있다. 이와 관련한 분야 중 커밋 메시지를 소프트웨어 유지관리의 유형으로 분류하는 많은 연구가 있으며 연구 중 최대 정확도는 87%다. 본 논문에서는 커밋 분류 모델을 이용한 솔루션 등의 활용을 목적으로 진행 하였고 기존에 발표된 모델들보다 정확도를 높여 모델의 신뢰성을 높이기 위해 여러 모델을 조합한 복합 분류 모델을 설계하고 구현하였다. 본문은 자동화 레이블링 및 소스 변경 내용을 추출하여 데이터셋을 구성하고 디스틸 버트(DistilBERT) 모델을 이용하여 학습시켰다. 검증결과 기존 연구에서 보고된 최대 87%보다 8%가 향상된 95%의 F1 점수 값을 얻어 신뢰성을 확보하였다. 본 연구 결과를 이용하면 모델의 신뢰성을 높이고 이를 이용해 소프트웨어 및 프로젝트관리 등의 솔루션에 적용이 가능할 것으로 기대된다.

▸ **주제어**: 커밋 메시지, 다중 레이블 분류, 디스틸 버트, 소스 변경, 자동 레이블링

• First Author: Ji-Hoon Choi, Corresponding Author: Joon-Yong Kim
*Ji-Hoon Choi (hunnx27@gmail.com), Dept. of Computer Engineering, Kongju National University
**Joon-Yong Kim (jykim@kongju.ac.kr), Dept. of IT Convergence Software, Seoul Theological University
*Seong-Hyun Park (a94270816@gmail.com), Dept. of Computer Engineering, Kongju National University
• Received: 2022. 08. 31, Revised: 2022. 11. 07, Accepted: 2022. 11. 07.
• This paper is an extension of the paper ("Proposal of Git's Commit Message Complex Classification Model for Effective S/W Maintenance") presented at the 66th Summer Conference of the Korera Computer Information Society in 2022.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

소스 버전 관리를 위한 SCM(Source Control Management)의 한 종류인 Git은 2000년대 초 리눅스 커널 관리를 위해 Linus Torvalds가 개발한 시스템으로 현재 많은 개발자들이 프로젝트 진행과 유지보수 활동에 이용하고 있다.

이러한 Git은 여러 사용자들 사이에 발생하는 소스코드의 변경 내용과 이슈를 관리하는데 이때 한 번의 변경 단위를 커밋이라고 부른다.

프로젝트 기간에는 개발단계와 단위테스트 단계에서 신규 기능 및 단위테스트 소스가 커밋 단위로 순차 저장되고, 프로젝트 종료 후에는 유지보수관리 활동을 진행하면서 개선 및 버그사항에 대한 조치 내용이 저장되기 때문에 커밋의 이력은 프로젝트와 밀접한 관련이 있다.

그래서 프로젝트와 운영활동의 리스크 예상과 프로젝트의 현황을 파악하고 이를 통해 비용과 시간에 대한 효율을 향상하기 위해 많은 연구자들이 방대한 양의 커밋 메시지를 분석하고 있다[1].

이런 목적의 연구의 한 분야로 커밋 메시지를 소프트웨어 유지보수 관리 행위의 세 가지 유형으로 분류하는 연구가 있다[2].

본 논문에서는 커밋 분류 모델을 이용한 솔루션 등의 활용을 목적으로 진행하였다. 솔루션 등은 비용과 밀접하게 관련이 있기 때문에 신뢰성과 정확도에 민감하기 때문에 기존에 발표된 분류 모델들보다 정확도를 높여 모델의 신뢰성을 높임을 목표로 연구를 진행하였다. 정확도를 높이기 위해 기존 제안했던 모델들을 분석하고 그 중 정확도가 높은 모델들의 장점들을 조합한 커밋 메시지 복합 분류 모델을 구현하여 해당 과정을 설명하고 정확도가 향상됨을 검증함으로써 관련 모델을 이용한 프로젝트 관련 시스템에 도입이 가능함을 입증하였다.

II. Preliminaries

1. Related works

1.1 Software Maintenence Type

다음 표 1은 소프트웨어 유지관리를 세 가지 형태로 분류한 것이다[2].

Table 1. Software Maintenance Type

Type	Description
Corrective	Fix Bug
Perfective	Performance Improvement
Adaptive	Feature Addition

소프트웨어 유지관리의 활동은 다음과 같다.

첫째, Corrective은 문제가 되는 기능 및 비기능에 대해 버그를 조치하는 활동이다.

둘째, Perfective은 시스템 및 디자인의 보완을 위해 개선하는 활동이다.

셋째, Adaptive은 신규 기능을 도입하거나 추가하는 활동이다. 신규 프로젝트 개발 단계에서 주로 발생된다.

본 논문에서는 커밋 메시지를 위에서 설명한 세 가지 유지관리 유형으로 분류하고 커밋과 소프트웨어 유지관리의 연관관계를 형성해주는 모델의 분류 정확도를 높이기 위한 연구를 진행했다.

1.2 Commit Message Classification Model By Utilizing Source Code Change

다음 그림 1은 소스 변경 데이터를 이용한 분류 모델의 프로세스이다.

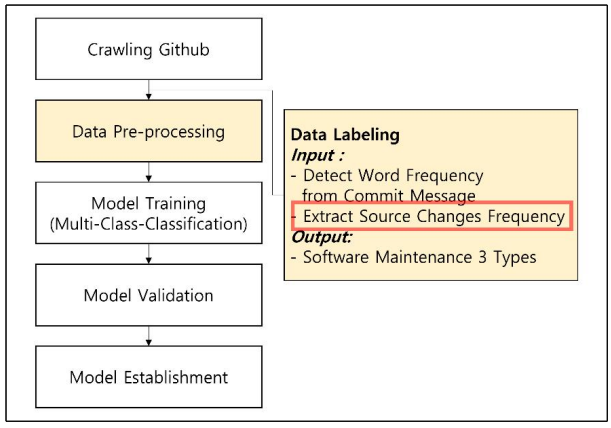

Fig. 1. Classification Process Using Source Change

이 분류 모델은 유지관리 분류마다 가장 많이 등장하는 핵심 키워드를 추출하고 커밋 메시지에서 빈도 배열을 만드는 역할을 수행한다. 또한 Fluri의 소스 변경 추출 도구인 체인지디스틸러(ChangeDistiller)를 이용해 소스 변경 유형을 추출하여 빈도 배열을 만들어 모델의 데이터로 이용한다[6-9]. 객관적인 데이터를 이용하여 정확도(Accuracy)는 76%까지 향상 할수 있었다[3].

하지만 소스 변경 유형의 데이터를 추출하기 위해서는 커밋에서 변경된 소스목록과 그 이전 버전의 실제 파일들

이 필요한데 해당 분류 모델은 유형 추출을 위해 커밋 별 모든 프로젝트의 버전을 모두 복사하여 이용한다. 이러한 과정은 단순 모델의 성능 실험에는 유효하지만 사용에는 어려움이 많다.

본 논문에서는 소스 변경 유형의 데이터의 이용을 위해 해당 과정을 Git CLI(Command Line Interface)를 이용해 해당 과정을 간소화하고 필요한 소스 파일만을 생성하고 데이터 추출 과정을 배치로 개발하여 자동화시켰다.

해당 내용은 본문의 자동화 레이블링 프로세스에서 자세히 다룬다.

1.3 Commit Message Classification By Utilizing DistilBERT

다음 그림 2는 디스틸 버트(DistilBERT) 전이학습 모델을 이용한 분류 모델이다.

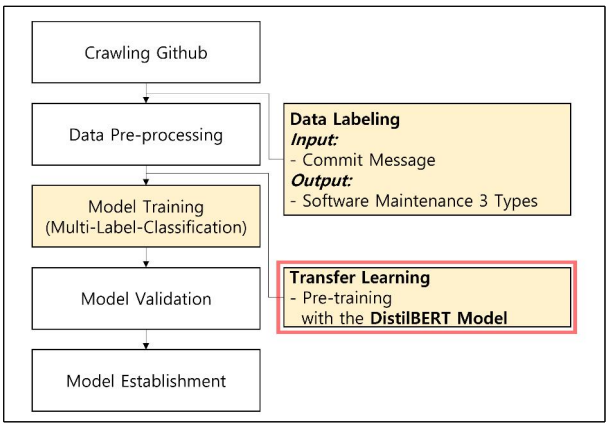

Fig. 2. Classification Process Using DistilBERT Model

이 연구에서는 버트(BERT)보다 빠르고 가벼운 디스틸 버트 전이학습 모델을 선택하여 커밋 메시지를 분류하였다[5][10].

또한 기존의 커밋 분류 모델이 다중(Multi-Class) 분류 모델로 유형 중 하나만 선택하는 분류 모델이었다면 해당 연구는 다중 레이블(Multi-Label) 분류 모델로 선택하였다. 디스틸 버트를 이용해 사전 훈련시키고 커밋 메시지를 분류했을 때 F1 점수는 87%까지 향상 됐다[11-12].

1.4 Source Change Type By ChangeDistiller

다음 표 2는 Fluri 연구에서 개발한 체인지디스틸러(ChangeDistiller)에서 제공하는 소스 변경에 대한 48가지 유형 중 일부 예제이다[6-9].

Table 2. Source Change Type

Idx	Type
3	ADDITIONAL_CLASS
4	ADDITIONAL_FUNCTIONALITY
5	ADDITIONAL_OBJECT_STATE
6	ALTERNATIVE_PART_DELETE
7	ALTERNATIVE_PART_INSERT
9	ATTRIBUTE_TYPE_CHANGE
10	CLASS_RENAMING
21	METHOD_RENAMING
22	PARAMETER_DELETE
23	PARAMETER_INSERT
25	PARAMETER_RENAMING
33	REMOVED_CLASS
34	REMOVED_FUNCTIONALITY
42	STATEMENT_DELETE
41	STATEMENT_INSERT

체인지디스틸러는 소스 파일을 비교해서 변경된 부분을 표 2와 같이 48가지의 유형으로 차이점 정보를 제공한다. 해당 유형의 데이터의 빈도수는 본 논문에서 모델 학습 시 커밋 메시지와 같이 들어가는 입력 데이터이다. 개발자가 직접 입력하는 커밋 메시지는 주관적이고 부정확할 수 있기 때문에 객관적 정보인 소스 변경의 데이터를 이용하여 모델의 정확도를 높였다.

III. The Proposed Scheme

1.1 System Configuration

다음 그림 3은 본 논문에서 구성한 복합 커밋 분류 모델의 시스템 구성도이다.

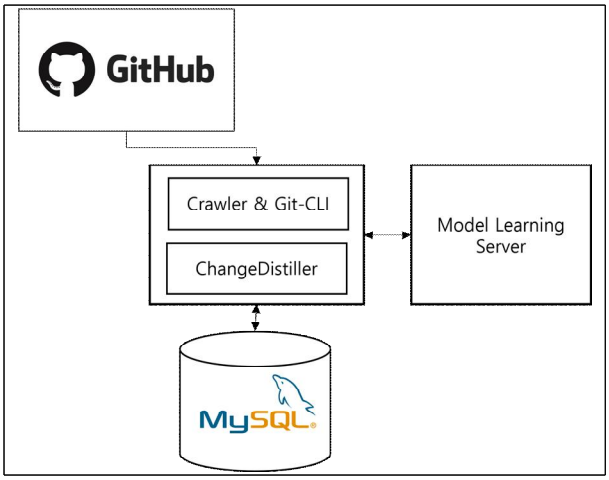

Fig. 3. System Configuration

시스템구성은 크롤러 모듈과 소스 변경을 추출하는 모듈이 있는 전처리 서버로 JAVA 기반의 스프링프레임워크로 구성되어 있고 본 논문의 1.3.1 Auto Labeling단원과 1.3.2 Manual Labeling단원의 프로세스를 수행한다. 깃 프로젝트에서 데이터 크롤링 및 데이터 추출을 수행하고 추출된 데이터를 통해 데이터 레이블링 작업을 진행한다, 모델 학습 서버에서는 파이썬 언어로 구성하여 데이터 학습을 진행한다. Pandas, numpy, simpletransformers의 라이브러리를 이용해 데이터 학습 및 검증을 수행하고, Seaborn, tensorboard 라이브러리를 이용해 학습 수행 과정과 학습 결과를 히트맵과 선형차트로 표현한다.

1.2 Target Projects

분석 대상 프로젝트는 다음 표 3과 같다.

Table 3. Target Projects

No	Project Name	Star	Fork
1	elastic/Elasticsearch	61.4k	22.3k
2	oracle/Graal	17.6k	1.4k
3	aosp-mirror/Platform Frameworks Base	10.1k	6.2k

우선, 분석할 프로젝트는 체인지디스틸러의 이용이 JAVA 기반의 프로젝트에만 적용할 수 있기 때문에 참조한 두 가지 모델에서 선정한 프로젝트의 목록이면서 JAVA를 기반으로 하는 프로젝트를 선택하였다.

세 개의 모든 프로젝트는 모두 Github에서 공개되어 있는 오픈소스이며 모두 Github에서 해당 프로젝트 이름으로 검색이 가능하다. Star는 프로젝트를 추천하는 수를 말하며, Fork는 해당 프로젝트를 기반으로 하는 프로젝트의 수를 말한다. 본 논문에서 선정한 세 개의 프로젝트는 모두 10000건 이상의 개발자가 추천한 프로젝트이며 이를 기반으로 하는 소스는 1000건 이상으로 많은 사용자가 이용하며 많은 프로젝트가 참조하고 있기 때문에 신뢰 높은 프로젝트임을 알 수 있다.

선정된 세 개의 프로젝트에서는 커밋 데이터 학습을 위해 프로젝트마다 1000건 씩 총 3000건의 데이터를 추출하였다.

1.3 System Process

다음 그림 4는 구현 모델의 시스템 프로세스이다.

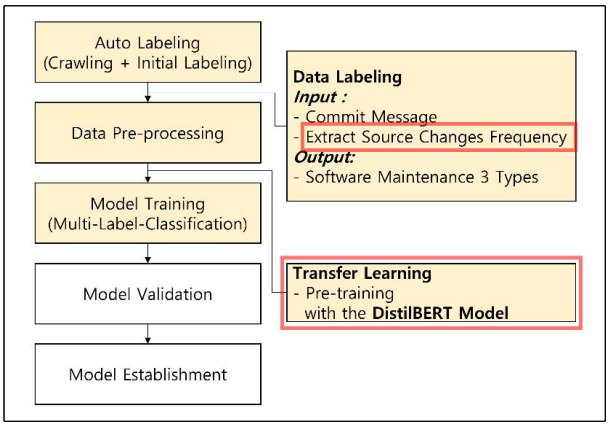

Fig. 4. Proposed Classification Process

전체적인 프로세스는 자동화 레이블링과 수동 레이블링, 데이터 전처리, 전이학습, 모델검증, 모델 수립의 순서로 진행된다. 단계에 대한 상세한 내용은 다음과 같다.

1.3.1 Auto Labeling

다음 그림 5는 자동화 레이블링 프로세스이다.

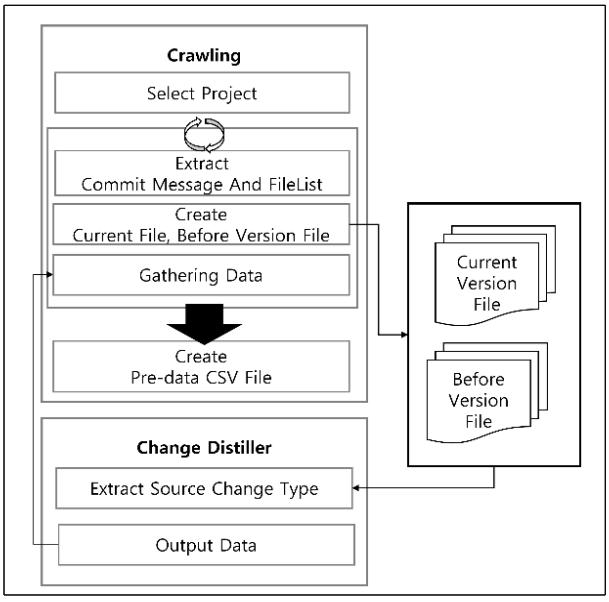

Fig. 5. Auto Labling Process

자동화 레이블링 프로세스는 크게 크롤링 단계와 소스 변경 추출 단계 그리고 데이터 취합 단계로 구성된다.

크롤링 단계에서는 깃헙(Git-Hub)에서 복사(Clone)한 프로젝트에서 Git CLI(Command Line Interface)를 이용해 커밋 메시지와 해당 커밋의 파일 목록을 추출하고, 커밋 시점의 소스 파일과 이전 버전의 소스 파일을 파일로 저장한다[15-16].

소스 변경 추출 단계에서는 생성된 소스 파일과 이전 버전 파일을 체인지디스틸러 프로그램을 통해 변경사항을 추출한다.

데이터 취합 단계에서는 이전 단계에서 추출한 커밋 메시지와 소스 파일 변경 유형을 취합하고 유형 별 빈도가 가장 많은 키워드를 이용해 키워드가 존재하는 커밋 메시지에 최초 자동 레이블링을 입력한다.

1.3.2 Manual Labeling

다음 그림 6은 1.3.1 Auto Labeling 단원의 과정을 통해 만들어진CSV(Comma-Seperated Values)데이터의 일부이다.

commitMsg	changeType	Corrective	Perfective	Adaptive
add file delete retry to testcluster elasticsearchnode (#89095) add retry logic for cleanup / deletion in testcluster's elasticsearchnode, to tolerate the asynchronous nature of deletions on the windows file-system.	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	0	0	1
fix object equals for sqlqueryrequest's binarycommunication (#87887) co-authored-by: owenniceliu <owenniceliu@tencent.com>	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]	1	0	0
...	...	...	...	...

Fig. 6. Labeled Data

commitMsg는 추출한 커밋 메시지이고, changeType은 관련연구 1.4 Source Change Type By ChangeDistiller 단원에서 설명한 체인지디스틸러를 이용한 소스변경 유형의 인덱스이다. 그리고 3~5열의 Corrective,Perfective,Adaptive는 관련연구 1.1 Software Maintenence Type에서 설명한 소프트웨어 유지관리의 세가지 형태이고 해당하는 경우 1, 해당하지 않는 경우0으로 레이블링한 데이터이다.

총 3000건의 데이터는 본 과정에서 업무 담당자들이 직접 추가적인 수동 레이블링 작업을 진행한다.

먼저 담당자의 판단으로 잘못 분류된 데이터는 바르게 수정하고 깃에서 머지같은 과정에서 생긴 자동 생성된 데이터는 의미가 없는 고정된 시스템 메시지기 때문에 삭제하였다. 수동 레이블링 작업 후 최종 데이터는 2162건이 남았다.

1.3.3 Data Pre-Processing

다음 그림 7은 전처리 과정의 소스 코드이다.

```
f = open("all.csv", "r")
df = pd.read_csv(f)
df['labels']=₩
  df[['T1','T2','T3']]₩
  .values.tolist()
df['text'] =₩
  "[CLS] "₩
  + df['commitMsg']₩
  + " [SEP] "₩
  + df['changeType']₩
  + " [SEP]"
df=df[['text','labels']]
train, test = train_test_split(df, test_size=0.20)
train.head(3)
```

Fig. 7. Pre-Processing Source Code

디스틸 버트 모델은 사전 학습 모델이기 때문에 대량의 단어 임베딩 등에 대해 사전 학습이 되어 있는 모델을 제공하기 때문에 데이터 전처리에 대부분의 처리를 생략할 수 있으므로 데이터 전처리 과정은 일반적인 분류 모델보다 간소화된다.

전처리 과정에서는 디스틸 버트로 학습할 수 있도록 프로토콜 규격에 맞게 데이터를 셋팅해준다.

첫째, 레이블 데이터는 'labels' 칼럼에 리스트 형태로 변환 시켜준다.

둘째, 제안 모델의 입력 데이터가 2가지 형태이므로 '[SEP]' 이라는 구분자로 커밋 메시지와 데이터 변경 유형의 빈도 배열을 구분하여 디스틸 버트모델의 프로토콜에 맞게 변환한다[11-12].

셋째, 전처리된 데이터를 학습 데이터와 검증 데이터로 8:2 비율로 나눈다. 검증 시 학습 데이터와는 다르게 검증 데이터를 분리해야 과적합(Overfitting)을 방지할 수 있다.

1.3.4 Transfer Learning

디스틸 버트 모델에 최적의 전이학습을 위해 하이퍼파라미터(Hyper Parameter)를 셋팅한다. 도출된 파라미터는 다음 표 4와 같다.

Table 4. Hyper Parameter Config Values

Config	Value
num_train_epochs	10
learning_rate	2e-05
train_batch_size	8

학습시 epoch수는 10회 설정을 넘어가면서 학습정확도가 떨어짐을 확인하고 10회로 정의하였고, learning rate는 학습을 반복하면서 파라미터 조정 빈도를 말하는데 BERT[11] 논문에서는 BERT 모델을 이용할 때 learning rate값이 5e-5, 4e-5, 3e-5, 2e-5에서 최고의 파인튜닝 값이라고 말한다. 4개의 설정값을 적용해본 결과 2e-5의 값일 때 가장 최적의 학습 곡선을 그렸다. 또한 batch size는 32로 설정할 때 최적이라고 보고 됐지만 본 논문의 실험에서는 8까지 낮췄을 때 최적의 학습 곡선을 나타냄을 확인하였다.

다음 그림 8은 학습이 진행되는 모습이다.

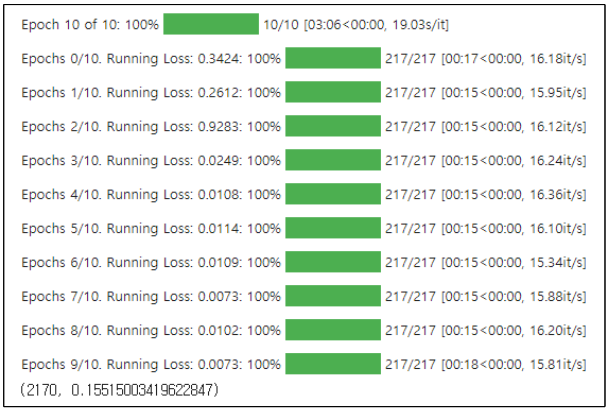

Fig. 8. Learning Progress

학습은 설정에 따라 10회의 반복 학습을 진행한다. 총 2162건의 데이터를 배치(Batch) 크기인 8로 나누어 1회의 에포크(Epoch)마다 217개의 스텝(Step)으로 이루어져 결과적으로 2170회의 반복(Iteration)을 통해 파라미터를 업데이트 한다.

다음 그림 9는 학습 시 손실률과 학습률 그래프이다.

10회의 에포크 동안 학습이 계속될 수록 손실률은 떨어져서 최종 학습 시에는 0.0073까지 떨어짐을 확인할 수 있다. 또한, 학습률은 설정대로 2e-5까지 선형적으로 증가 이후 0에 수렴하는 모습을 보인다. 이는 해당 모델이 신뢰할 수 있는 모델임을 확인할 수 있다.

1.3.5. Model Validation & Evaluation

학습이 완료되면 20%의 검증데이터로 모델을 검증하고 결과값을 확인한다. 다중 레이블 분류이기 때문에 정밀도(Percision)과 재현율(Recall) 결과를 추출하고 이를 이용한 F1 점수를 도출하였다.

다음 그림 10은 모델 검증 결과이고, 표 5는 제안 모델의 정확도와 다른 모델의 정확도를 비교한 표이다.

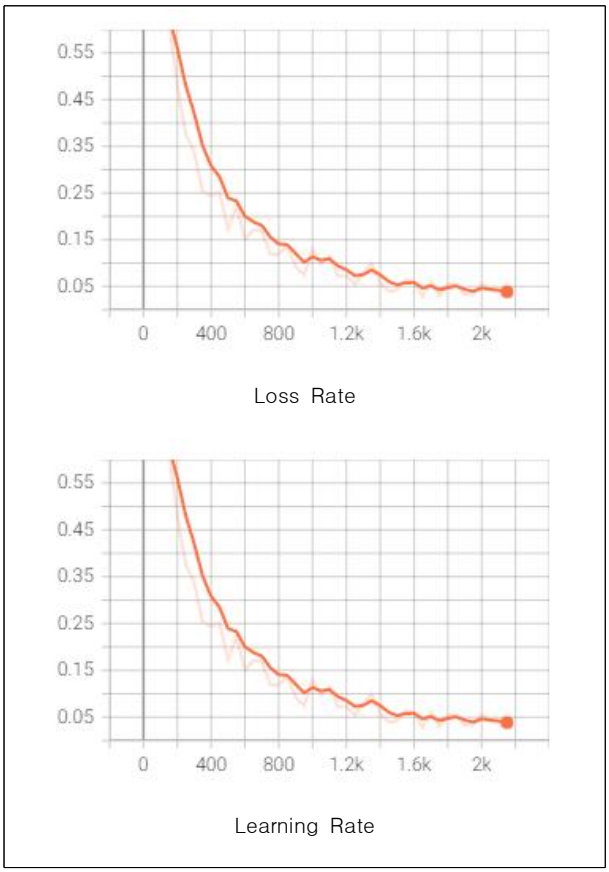

Fig. 9. Loss & Learning Rate

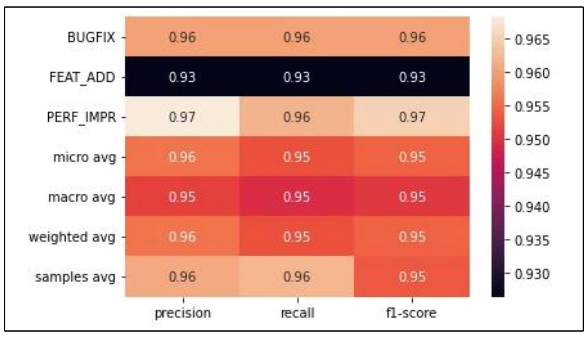

Fig. 10. Model Evaluation

F1 점수는 정밀도와 재현율의 평균으로 정의된다. F1 점수 범위는 높을수록 좋다. 제안 모델의 검증 결과는 기존 연구에서 제시했던 모델보다 가장 높은 F1 점수인 95%의 결과를 보여준다.

Table 5. Compare Model Results

Dataset	F1-Score	H-Loss
Proposal Model	**95%**	**0.04**
M. U. Sarwar et al.	87%	0.11
S. Levin et al.	76%	-
Gharbi et al.	71%	0.22
Mauczka et al.	50%	0.31

M. U. Sarwar 외 4명의 연구인 디스틸 버트만 이용했던 모델보다 8% 향상됐고, S. Levin 외 1명의 연구인 키워드 빈도와 소스 변경 유형 빈도를 이용한 모델보다 19% 향상됐다. 기타 다른 커밋 분류 모델인 Gharbi 외 3명의 연구와 A. Mauczka 외 3명의 연구와 비교했을 때는 최대 45% 향상된 결과이다[13-14]. 이는 제안한 복합 분류 모델이 위에서 비교한 4개의 논문의 모델의 정확도 및 F1점수보다 최소8%, 최대45%를 향상시킴을 확인할 수 있다.

IV. Conclusions

본 논문은 기존 연구의 커밋 분류 모델의 정확도가 실제 프로젝트 적용에 어려움을 느끼고 정확도를 향상하기 위해 가장 높은 정확도와 F1 점수를 가진 분류 모델을 선정하여 모델을 분석하고 장점을 추출하여 복합 분류 모델을 설계하였고 실제 구현 및 모델을 검증하였다.

검증 결과 F1 점수는 95%로 상당히 개선된 모델임을 검증하였다. 또한 95%의 점수로 상당히 신뢰가 높은 점수를 바탕으로 구현된 모델을 이용한 프로젝트 관리에 도움이 될 수 있는 일감관리시스템에 적용하여 자동으로 일감과 관련된 커밋메시지의 후보 목록을 제안할 수 있는 시스템을 설계하고자 한다. 또한 모델의 신뢰성을 높이고 이를 이용해 소프트웨어 및 프로젝트관리 등의 솔루션에 적용이 가능할 것으로 기대된다.

REFERENCES

[1] Mockus and Votta, "Identifying reasons for software changes using historic databases," Proceedings 2000 International Conference on Software Maintenance, pp. 120-130, 2000. doi: 10.1109/ICSM.2000.883028.

[2] S. Gharbi, M. W. Mkaouer, I. Jenhani, and M. B. Messaoud, "On the Classification of Software Change Messages Using Multi-Label Active Learning," in Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, 2019, pp. 1760-1767. 2019. doi: 10.1145/3297280.3297452

[3] S. Levin, and A. Yehudai, "Boosting Automatic Commit Classification Into Maintenance Activities By Utilizing Source Code Changes," PROMISE: Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering, pp. 97-106, November. 2017. doi: 10.1145/3127005.3127016

[4] A. Adhikari, A. Ram, R. Tang, and J. Lin, "DocBERT: BERT for Document Classification," arXiv, 2019. doi: 10.48550/ARXIV.1904.08398

[5] M. U. Sarwar, S. Zafar, M. W. Mkaouer, G. S. Walia and M. Z. Malik, "Multi-label Classification of Commit Messages using Transfer Learning," 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 37-42, 2020, doi: 10.1109/ISSREW51248.2020.00034.

[6] H. C. Gall, B. Fluri, and M. Pinzger, "hange analysis with evolizer and changedistiller," IEEE Software, vol. 26, no. 1, p. 26, 2009. DOI: 10.1109/MS.2009.6

[7] B. Fluri, E. Giger, and H. C. Gall, "Discovering patterns of change types," in Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on. IEEE, 2008, pp. 463-466. DOI: 10.1109/ASE.2008.74

[8] M. Martinez, L. Duchien, and M. Monperrus, "Automatically extracting instances of code change patterns with ast analysis," arXiv preprint arXiv:1309.3730, 2013. DOI: 10.1109/ICSM.2013.54

[9] B. Fluri, M. Wursch, M. PInzger, and H. C. Gall, "Change distilling: Tree differencing for fine-grained source code change extraction," Software Engineering, IEEE Transactions on, vol. 33, no. 11, pp. 725-743, 2007. DOI: 10.1109/TSE.2007.70731

[10] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019. DOI: 10.48550/arXiv.1910.01108

[11] Devlin, Jacob, et al. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, June 2019, pp. 4171-4186. DOI:10.18653/v1/N19-1423

[12] Sun, Chi, et al. "How to Fine-Tune BERT for Text Classification?" Lecture Notes in Computer Science, 2019, pp. 194-206. DOI:10.1007/978-3-030-32381-3_16

[13] A. Mauczka, F. Brosch, C. Schanes, and T. Grechenig, "Dataset of developer-labeled commit messages," in 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE, 2015, pp. 490-493. DOI: 10.1109/MSR.2015.71

[14] S. Zafar, M. Z. Malik, and G. S. Walia, "Towards standardizing and improving classification of bug-fix commits," in 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, 2019, pp. 1-6. DOI: 10.1109/ESEM.2019.8870174

[15] Hultstrand, S., & Olofsson, R. (2015). Git-CLI or GUI: Which is most widely used and why?.

[16] Miller, C. G. (2022). Introduction to Git.

Authors

Ji-Hoon Choi received his master's degree in computer Engineering from Kongju University in March 2020. Currently, the doctoral program in Computer Science and Engineering at Kongju University is in progress from March 2021. Ji-Hoon Choi started programming as a Web-based JAVA program in 2014. Currently, as a manager at Ice Cream Kids Company, he is in charge of developing and operating B2C services and LCMS related to Early-childhood-education. He is interested in project management systems, automated testing, algorithms, machine learning, and automated systems.

Joon-Yong Kim received the B.S.,degrees in Civil Engeneering from SungKyunKwan University, Korea, in 1985. Then received the M.S. and Ph.D. degrees in Computer Science and Engineering from KongJu National University, Korea, in 2013 and 2018, respectively. Dr. Kim joined the faculty of the Department of IT Convergence Software at Seoul Theological University, Gyeonggi-do, Korea, in 2020. He is currently a Professor in the Department of IT Convergence Software, Seoul Theological University. He is interested in Machine Learning, Auto ML, and AI.

Seong-Hyun Park received the B. S. degree in College of Arts and Music from Chungnam National University, Korea in 2017. The M. S, and Ph. D. degrees in Computer Engineering from Kongju National University, Korea, in 2017, 2020. He is currently a teaching in the Department of Computer Science & Engineering. Kongju National University. He is interested in computer music, convergence Education. and Real Time System and Management and Clout computing and Communication.