GPT를 이용한 Git의 커밋메시지 분류모델 제안

최지훈[O], 김재웅[**], 이윤열[*], 채의근[*], 서현호[*]
[O]공주대학교 컴퓨터공학과,
[*]공주대학교 컴퓨터공학과,
[**]공주대학교 소프트웨어학과
e-mail: hunnx27@gmail.com[O], jwkim@kongju.ac.kr[**],
{alphaone, ygchae}@kongju.ac.kr[*], shhmetis@naver.com[*]

Proposal of Git's commit message classification model using GPT

Ji-Hoon Choi[O], Jae-Woong Kim[**], Youn-Yeoul Lee[*], Yi-Geun Chae[*], Hyeon-Ho Seo[*]
[O]Dept. of Computer Engineering, Kongju National University,
[*]Dept. of Computer Engineering, Kongju National University,
[**]Dept. of Software, Kongju National University

● 요 약 ●

GIT의 커밋 메시지를 소프트웨어 유지보수 활동 세 가지로 분류하는 연구를 분석하고 정확도를 높일 수 있는 모델들을 분석하였고 관련 모델 중 커밋메시지와 변경된 소스를 같이 활용하는 연구들은 변경된 소스를 분석하기 위해 도구들을 대부분 활용하는데 대부분 특정 언어만 분류할 수 있는 한계가 있다. 본 논문에서는 소스 변경 데이터를 추출할 때 언어의 제약을 없애기 위해 GPT를 이용해 변경된 소스의 요약을 추출하는 과정을 추가함으로써 언어 제약의 한계를 극복할 수 있는 개선된 모델에 관한 연구를 진행하였다. 향후 본 연구 모델의 구현 및 검증을 진행하고 이를 이용해 프로젝트 진행에 활용할 수 있는 솔루션 개발 연구까지 확정해 나갈 예정이다.

키워드: 커밋 메시지(Commit Message),
BERT(Bidirectional Encoder Representations from Transformers),
GPT(Generative Pre-trained Transformer)

I. Introduction

소프트웨어 개발과 유지보수 활동을 위해 소스 코드의 형상 관리는 필요하다. 프로젝트의 유지보수 관리를 하면서 프로그램의 개선 행위, 버그 조치 행위, 신규 기능 추가와 같은 다양한 유지보수 활동이 이루어지고 이 과정에서 소스 코드가 변경이 되는데 어떻게 소스 코드가 변경됐는지 왜 변경됐는지 버전 관리를 해주는 것이 소스 코드 형상 관리이다. 이를 도와주는 도구로 2000년대 초에 리누스 토르발스(Linus Torvalds)가 개발한 깃(GIT)이란 도구가 현재 주류를 이루고 있다.

깃의 도구에서는 소스 코드를 변경하고 버전을 변경하는 시점을 커밋(Commit)이라는 용어로 부른다. 커밋이란 이전 버전부터 현재 적용하는 버전까지의 모든 변경된 소스 코드를 저장하고 커밋 메시지(Commit Message)를 작성해 변경된 이유와 무엇을 변경했는지 요약내용을 함께 저장한다.

저장된 커밋을 분석하면 프로젝트의 계획, 리스크 예측할 수 있고 이를 통해 비용을 줄이고 시간 효율을 향상할 수 있다[1]. 이러한 이유로 커밋을 분석하는 다양한 연구가 있고 커밋을 유지보수 활동 유형으로 분류하는 연구는 그러한 연구 중 하나이다.

지난 연구에서 다양한 분류 모델들[2-4]을 조합하여 복합 분류 모델을 설계 및 개발하여 정확도(F1-Score)는 95%까지 향상하고 손실율(H-Loss) 0.04 결과를 도출하였다[5]. 하지만 해당 연구에서는 소스변경 데이터를 추출할 때 사용한 도구는 특정 언어에 한정되어 있어서 해당 언어 외의 프로젝트 소스는 분류할 수 없는 한계가 있다.

본 논문에서는 특정 언어의 제약 없이 어떤 언어로 작성된 소스 코드라도 동일하게 소프트웨어 유지보수 활동 카테고리로 분류할 수 있는 분류 모델을 연구하고 분류 모델의 프로세스를 제안한다.

II. Preliminaries

1. Related works

1.1 소프트웨어 유지관리 세 가지 분류

다음 Table 1. 은 소프트웨어 유지관리를 세 가지 유형으로 분류한 것이다[6].

Table 1. Software Maintenance Category

Category	Action Summary
Corrective	Software Bug Fix Action
Perfective	Software Improvement Action
Adaptive	The act of introducing a new function to the software

코렉티브(Corrective 유형은 소프트웨어에서 문제가 되는 버그를 고치는 조치 행위로 분류되는 커밋 유형이고,

퍼펙티브(Perfective) 유형은 소프트웨어를 보완하기 위해 개선하는 행위로 분류되는 커밋 유형이다.

마지막으로, 어댑티브(Adaptive) 유형은 소프트웨어에 새로운 기능을 도입하는 행위로 분류되는 커밋 유형이다.

본 논문에서는 커밋 메시지를 위와 같은 세 가지 유지관리 활동으로 분류하고 분류 정확도를 높이기 위한 방법과 언어 제약 없이 소스 코드 분류가 가능한 모델 연구를 진행하였다.

III. The Proposed Scheme

본 논문에서는 커밋 메시지를 분류하는 다양한 연구를 분석하고 자연어 생성 모델인 GPT(Generative Pre-trained Transformer)를 이용해 특정 언어의 제약을 없애는 분류 모델을 설계하였다. 분류 모델의 프로세스는 다음 Fig. 1. 과 같다.

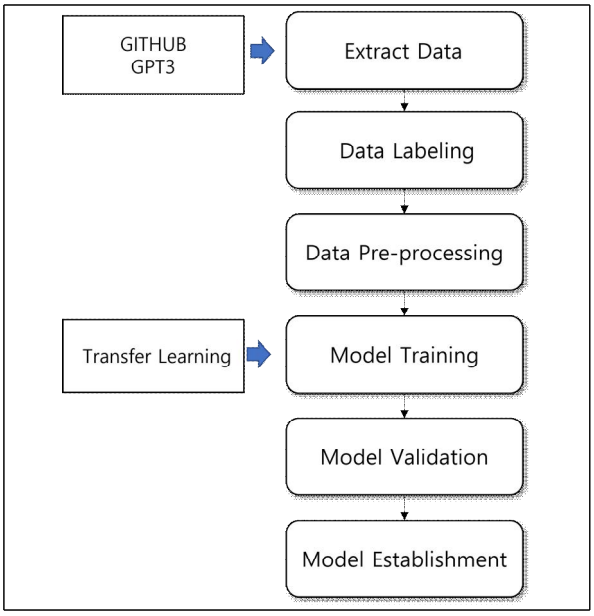

Fig. 1. Proposed Classification Process

첫째, 깃헙(Github)에 공개된 신뢰 높은 프로젝트를 선별한다. 프로젝의 선별 기준은 깃헙에서 제공하는 프로젝트의 평점이 높고 프로젝트를 이용 수가 많은 프로젝트를 선별한다. 이용 수는 깃헙에서 포크(Fork)한 수가 많은 프로젝트를 기준으로 한다. 이전 연구에서 자바(JAVA) 언어에 한정된 소스 코드 분류만 가능했지만 제안하는 모델은 자바 언어 외 어떠한 언어로 작성된 소스 코드도 분류가 가능하므로 비교를 위해 이전 연구에서 선정했던 자바로 작성된 프로젝트를 포함해서 자바 언어가 아닌 다른 언어로 작성된 프로젝트도 추가 선택한다.

둘째, 선별된 프로젝트를 크롤링하여 프로젝트의 커밋 메시지 목록과 커밋에 포함된 소스 코드를 수집한다.

셋째, 배치코드를 이용해 커밋에 포함된 소스 코드를 GPT를 이용해 소스 코드 요약 내용을 추출한다. 이 과정에서 GPT에 다양한 형태의 질의를 실험하고 원하는 응답값이 나올 수 있도록 질문 템플릿을 정의한다.

넷째, 커밋 메시지와 GPT를 통해 추출된 소스 코드를 전 처리하고 세가지 소프트웨어 유지관리 활동 유형으로 레이블링(Labeling)한다.

다섯째, 데이터셋을 훈련세트와 검증세트를 8:2로 분류하여 과적합 현상을 방지한다.

여섯째, 버트(BERT:Bidirectional Encoder Representations From Transformers)모델을 이용해 훈련세트를 훈련시켜 분류 모델을 만들고 검증세트를 이용해 분류 정확도를 측정한다.

일곱째, 이전 분류 모델과의 정확도를 비교하여 제안한 분류 모델의 정확도를 비교하여 정확도의 차이를 비교하고, 자바 외의 프로젝트에서 동일하게 분류가 가능한지 검증한다.

IV. Conclusions

본 논문은 깃의 커밋을 소프트웨어 유지관리 활동으로 분류하는 연구를 진행하였다.

기존 연구에서는 제시한 커밋 분류 모델을 실제 프로젝트에 활용하기에는 정확도가 떨어지는 문제점을 개선하기 위해 최소 90% 이상의 분류 정확도를 구현할 수 있게 모델을 설계하였지만, 특정 언어로 작성된 소스 코드만이 분류가 가능한 한계가 있어 언어 제약을 한계를 극복할 수 있는 모델의 연구를 진행하였다.

이를 개선하기 위해 기존 프로세스에서 자바 코드를 분석해서 소스 변경 데이터를 추출했던 과정을 제거하고 GPT를 이용해 소스변경 내용 요약을 추출하는 과정으로 대체함으로써 특정 언어만 분류할 수 있었던 문제점을 개선할 수 있다.

향후 본 연구를 실제 구현하고 모델을 검증하고 이를 이용해 소프트웨어 프로젝트 진행에 활용할 수 있는 솔루션 개발 연구까지 확장해 나갈 예정이다.

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. 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

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

[4] 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.

[5] J,H. Choi, J.Y. Kim, and S. H. Park, "Implementation of Git's Commit Message Complex Classification Model for Software Maintenance," Journal of The Korea Society of Computer and Information Vol. 27, No. 11, pp. 131-138, 2022.

[6] 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