(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0091993
(43) 공개일자 2025년06월23일

(51) 국제특허분류(Int. Cl.)
G06V 30/413 (2022.01) *G06F 17/15* (2006.01)
G06V 10/20 (2022.01) *G06V 10/44* (2022.01)
G06V 10/46 (2022.01) *G06V 10/80* (2022.01)
G06V 30/16 (2022.01) *G06V 30/18* (2022.01)
G06V 30/182 (2022.01)
(52) CPC특허분류
G06V 30/413 (2022.01)
G06F 17/153 (2013.01)
(21) 출원번호 10-2023-0182517
(22) 출원일자 2023년12월14일
심사청구일자 2023년12월14일

(71) 출원인
경상국립대학교산학협력단
경상남도 진주시 진주대로 501 (가좌동)
(72) 발명자
이선아
경상남도 진주시 진주대로 501, 407동 307호
곽창원
경상남도 진주시 진주대로 501, 407동 301호
정필수
경상남도 진주시 진주대로 501, 407동 311호
(74) 대리인
특허법인더웨이브

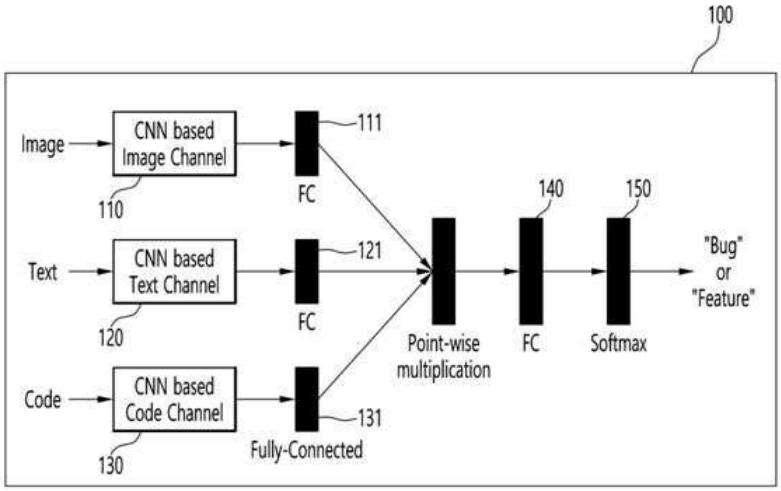
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 **버그 보고서의 멀티 모달 데이터를 처리하는 방법 및 장치**

(57) 요 약

일 측면에 버그 보고서를 분류하는 방법은, 소프트웨어의 버그와 관련된 버그 보고서에서 추출된 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성하는 단계; 상기 버그 보고서에서 추출된 코드 데이터를 제2 모달 모델을 통해 처리함으로써 제2 벡터를 생성하는 단계; 상기 버그 보고서에서 추출된 이미지 데이터를 제3 모달 모델을 통해 처리함으로써 제3 벡터를 생성하는 단계; 및 상기 제1 벡터, 상기 제2 벡터 및 상기 제3 벡터를 이용하여 상기 버그 보고서를 분류하는 단계;를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06V 10/20 (2023.08)
G06V 10/454 (2023.08)
G06V 10/469 (2023.08)
G06V 10/806 (2023.08)
G06V 30/16 (2023.08)
G06V 30/18057 (2023.08)
G06V 30/1823 (2023.08)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711184357
과제번호 2021R1A2C1094167
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 개인기초연구(과기정통부)
연구과제명 안전한 소프트웨어 변경을 위한 추적성의 문맥적 가시화 연구
기 여 율 60/100
과제수행기관명 경상국립대학교
연구기간 2023.03.01 ~ 2024.02.29
이 발명을 지원한 국가연구개발사업
과제고유번호 1345370630
과제번호 LINC3.0-2023-11
부처명 교육부
과제관리(전문)기관명 한국연구재단
연구사업명 산학연협력고도화지원
연구과제명 3단계산학연협력선도대학육성사업(LINC3.0)(0.5)
기 여 율 40/100
과제수행기관명 경상국립대학교
연구기간 2023.03.01 ~ 2024.02.29

명 세 서

청구범위

청구항 1

소프트웨어의 버그와 관련된 버그 보고서에서 추출된 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성하는 단계;

상기 버그 보고서에서 추출된 코드 데이터를 제2 모달 모델을 통해 처리함으로써 제2 벡터를 생성하는 단계;

상기 버그 보고서에서 추출된 이미지 데이터를 제3 모달 모델을 통해 처리함으로써 제3 벡터를 생성하는 단계; 및

상기 제1 벡터, 상기 제2 벡터 및 상기 제3 벡터를 이용하여 상기 버그 보고서를 분류하는 단계;를 포함하는 버그 보고서를 분류하는 방법.

청구항 2

제 1 항에 있어서,

상기 제1 벡터를 생성하는 단계는,

상기 텍스트 데이터에 대한 전처리를 수행하는 단계; 및

상기 전처리된 텍스트 데이터를 제1 모달 모델을 통해 처리함으로써 제1 벡터를 생성하는 단계;를 포함하는 방법.

청구항 3

제 2 항에 있어서,

상기 텍스트 데이터에 대한 전처리를 수행하는 단계는.

상기 텍스트 데이터의 대소문자를 통일하는 단계;

상기 대소문자 통일된 텍스트 데이터를 분할하여 토큰화 (tokenization)하는 단계; 및

상기 토큰화된 텍스트 데이터에서 불용어(stop word)를 제거하는 단계를 포함하는, 방법.

청구항 4

제 1 항에 있어서,

상기 제1 벡터를 생성하는 단계는,

상기 제1 모달 모델에 포함된 적어도 하나의 커널을 이용하여, 상기 텍스트 데이터에 대한 합성곱 연산을 수행하여, 적어도 하나의 합성곱 특징 벡터를 추출하는 단계;

상기 적어도 하나의 합성곱 특징 벡터에 대해, 맥스 풀링(max pooling) 연산을 수행하고, 상기 맥스 풀링 연산의 결과들을 결합(concatenate)하여 단일 벡터를 생성하는 단계; 및

상기 단일 벡터를 제1 완결 연결 레이어(Fully Connected layer)에 입력하여 상기 제1 벡터를 생성하는 단계를 포함하는, 방법.

청구항 5

제 1 항에 있어서,

상기 제2 벡터를 생성하는 단계는,

상기 코드 데이터에 대한 전처리를 수행하는 단계; 및

상기 전처리된 코드 데이터를 제2 모달 모델을 통해 처리함으로써 제2 벡터를 생성하는 단계;를 포함하는 방법.

청구항 6

제 5 항에 있어서,

상기 코드 데이터에 대한 전처리를 수행하는 단계는,

상기 코드 데이터에 포함된 자연어를 제거하는 단계;

상기 자연어가 제거된 코드 데이터에서 코드의 구조를 나타내는 부분을 이스케이프 시퀀스(escape sequence)로 대체하는 단계;

상기 대체된 코드 데이터의 대소문자를 통일하는 단계; 및

상기 대소문자 통일된 코드 데이터를 토큰화하는 단계를 포함하는, 방법.

청구항 7

제 1 항에 있어서,

상기 제2 벡터를 생성하는 단계는,

상기 제2 모달 모델에 포함된 적어도 하나의 커널을 이용하여, 상기 코드 데이터에 대한 합성곱 연산을 수행하여, 적어도 하나의 합성곱 특징 벡터를 추출하는 단계;

상기 적어도 하나의 합성곱 특징 벡터에 대해, 맥스 풀링(max pooling) 연산을 수행하고, 상기 맥스 풀링 연산의 결과들을 결합(concatenate)하여 단일 벡터를 생성하는 단계; 및

상기 단일 벡터를 제2 완결 연결 레이어(Fully Connected layer)에 입력하여 상기 제2 벡터를 생성하는 단계를 포함하는, 방법.

청구항 8

제 1 항에 있어서,

상기 제3 벡터를 생성하는 단계는,

상기 이미지 데이터에 대한 전처리를 수행하는 단계; 및

상기 전처리된 이미지 데이터를 제3 모달 모델을 통해 처리함으로써 제3 벡터를 생성하는 단계;를 포함하는 방법.

청구항 9

제 8 항에 있어서,

상기 이미지 데이터에 대한 전처리를 수행하는 단계는,

상기 이미지 데이터에 포함된 모든 이미지의 크기를 동일한 크기로 크기 조절하는 단계;

상기 크기 조절된 이미지 데이터의 분포 및 스케일을 정규화하는 단계를 포함하는, 방법.

청구항 10

제 1 항에 있어서,

상기 제3 벡터를 생성하는 단계는,

상기 전처리된 이미지 데이터에 합성곱 연산 및 ReLU 연산을 수행하여, 합성곱 특징 벡터를 추출하는 단계;

상기 합성곱 특징 벡터에 대해, 맥스 풀링(max pooling) 연산을 수행하는 단계; 및

상기 맥스 풀링 연산의 결과를 제3 완결 연결 레이어(Fully Connected layer)에 입력하여 상기 제3 벡터를 생성하는 단계를 포함하는, 방법.

청구항 11

제 4 항, 제 7 항 및 제 10 항 중 어느 한 항에 있어서,

상기 제1 벡터를 생성하는 단계, 상기 제2 벡터를 생성하는 단계 및 상기 제3 벡터를 생성하는 단계 중 어느 한 단계는,

상기 단일 벡터에 ReLU 연산 및 선형 연산을 수행하여 상기 제1 벡터, 상기 제2 벡터 및 상기 제3 벡터 중 어느 하나를 생성하는 단계를 포함하는, 방법.

청구항 12

제 1 항에 있어서,

상기 제1 벡터, 상기 제2 벡터 및 상기 제3 벡터를 이용하여 상기 버그 보고서를 분류하는 단계는,

상기 제1 벡터, 상기 제2 벡터 및 상기 제3 벡터에 요소별 곱셈(element-wise multiplication)을 수행하여, 제4 벡터를 획득하는 단계; 및

상기 제4 벡터를 이용하여 상기 버그 보고서를 분류하는 단계를 포함하는, 방법.

청구항 13

제 12 항에 있어서,

상기 버그 보고서를 분류하는 단계는,

상기 제4 벡터를 최종 완전 연결 레이어에 입력하고, 상기 최종 완전 연결 레이어로부터 출력을 획득하는 단계; 및

상기 출력을 소프트맥스 레이어에 입력하고, 상기 소프트맥스 레이어로부터 상기 버그 보고서가 버그 또는 특징에 해당할 확률을 획득함으로써 상기 버그 보고서를 분류하는 단계를 포함하는, 방법.

청구항 14

적어도 하나의 메모리; 및

적어도 하나의 프로세서;를 포함하고,

상기 프로세서는,

소프트웨어의 버그와 관련된 버그 보고서에서 추출된 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성하고,

상기 버그 보고서에서 추출된 코드 데이터를 제2 모달 모델을 통해 처리함으로써 제2 벡터를 생성하고,

상기 버그 보고서에서 추출된 이미지 데이터를 제3 모달 모델을 통해 처리함으로써 제3 벡터를 생성하고,

상기 제1 벡터, 상기 제2 벡터 및 상기 제3 벡터를 이용하여 상기 버그 보고서를 분류하는, 버그 보고서를 분류하는 장치.

청구항 15

제 1 항에 따른 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

발명의 설명

기 술 분 야

[0001] 본 개시는, 소프트웨어의 버그와 관련된 버그 보고서를 분류하는 방법 및 장치에 관한다.

배 경 기 술

[0002] 소프트웨어 제품을 사용하는 동안 사용자는 이슈 보고서(이하, 버그 보고서)를 작성하여 버그, 기능 제안 및 기타 의견을 보고한다. 개발자는 사용자가 제출한 버그 보고서를 수집하여 소프트웨어의 문제를 감지할 수 있고, 버그 보고서를 참조하여 소프트웨어 제품을 변경할 수 있다. 그러나 버그 보고서의 수가 급격히 증가하고 있고, 활발히 활동 중인 오픈소스 소프트웨어 프로젝트의 경우 매일 수백 건의 버그 보고서가 누적되어, 이러한 버그 보고서를 수동으로 관리하는 것에는 개발자의 상당한 노력이 요구된다. 이에 따라 버그 보고서를 체계적으로 관리하기 위해 버그 보고서를 자동으로 분류하는 기술이 사용되어 왔다. 종래 기술은 버그 보고서에 기술된 텍스트 데이터만을 활용하는 단일 모달 모델에 의존한다. 반면에, 실제 버그 보고서에는 텍스트, 이미지 및 소스 코드를 비롯한 다양한 모달리티의 데이터가 존재하며, 이것들은 버그 보고서를 이해하고 분류하는데 도움을 주는 정보이다. 하지만 종래 기술은 이러한 정보를 활용하지 못하여 버그 보고서 분류 성능에 한계가 있다.

발명의 내용

해결하려는 과제

[0003] 멀티모달 딥러닝을 통해 이슈 보고서의 텍스트, 이미지 및 코드 정보를 결합함으로써 버그 보고서의 분류 작업을 고도화하고, 분류 성능을 개선할 수 있는 버그 보고서를 분류하는 방법 및 장치를 제공하고자 한다.

[0004] 버그 보고서에 포함된 텍스트, 이미지, 코드 등 다양한 유형의 정보를 이용하는 멀티 모달 딥러닝 기반의 버그 보고서 분류 방법 및 장치를 제공하는 데 있다. 또한, 상기 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공하는 데 있다. 해결하려는 기술적 과제는 상기된 바와 같은 기술적 과제들로 한정되지 않으며, 또 다른 기술적 과제들이 존재할 수 있다.

과제의 해결 수단

[0005] 일 측면에 따른 버그 보고서를 분류하는 방법은, 소프트웨어의 버그와 관련된 버그 보고서에서 추출된 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성하는 단계; 상기 버그 보고서에서 추출된 코드 데이터를 제2 모달 모델을 통해 처리함으로써 제2 벡터를 생성하는 단계; 상기 버그 보고서에서 추출된 이미지 데이터를 제3 모달 모델을 통해 처리함으로써 제3 벡터를 생성하는 단계; 및 상기 제1 벡터, 상기 제2 벡터 및 상기 제3 벡터를 이용하여 상기 버그 보고서를 분류하는 단계;를 포함한다.

[0006] 다른 측면에 따른 버그 보고서를 분류하는 장치는, 적어도 하나의 메모리; 및 적어도 하나의 프로세서;를 포함하고, 상기 프로세서는, 소프트웨어의 버그와 관련된 버그 보고서에서 추출된 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성하고, 상기 버그 보고서에서 추출된 코드 데이터를 제2 모달 모델을 통해 처리함으로써 제2 벡터를 생성하고, 상기 버그 보고서에서 추출된 이미지 데이터를 제3 모달 모델을 통해 처리함으로써 제3 벡터를 생성하고, 상기 제1 벡터, 상기 제2 벡터 및 상기 제3 벡터를 이용하여 상기 버그 보고서를 분류한다.

[0007] 또 다른 측면에 따른 컴퓨터로 읽을 수 있는 기록매체는 상술한 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 기록매체를 포함한다.

도면의 간단한 설명

[0008] 도 1은 일 실시예에 따른 버그 보고서를 분류하는 장치의 일 예를 설명하기 위한 도면이다.

도 2는 일 실시예에 따른 버그 보고서를 분류하는 장치의 일 예를 도시한 구성도이다.

도 3은 일 실시예에 따른 버그 보고서를 분류하는 방법의 일 예를 설명하기 위한 흐름도이다.

도 4는 일 실시예에 따른 버그 보고서의 일 예를 나타내는 도면이다.

도 5는 일 실시예에 따른 프로세서가 텍스트 데이터에 대한 전처리를 수행하는 일 예를 설명하기 위한 도면이다.

도 6은 일 실시예에 따른 프로세서가 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성하는 일 예를 설명하기 위한 도면이다.

도 7은 일 실시예에 따른 프로세서가 코드 데이터에 대한 전처리를 수행하는 일 예를 설명하기 위한 도면이다.

도 8은 일 실시예에 따른 프로세서가 이미지 데이터에 대한 전처리를 수행하는 일 예를 설명하기 위한

도면이다.

도 9는 일 실시예에 따른 프로세서가 이미지 데이터를 제3 모달(modal) 모델을 통해 처리함으로써 제3 벡터를 생성하는 일 예를 설명하기 위한 도면이다.

도 10은 일 실시예에 따른 프로세서가 제1 벡터, 제2 벡터 및 제3 벡터를 이용하여 버그 보고서를 분류하는 일 예를 설명하기 위한 흐름도이다.

도 11은 일 실시예에 따른 프로세서가 제4 벡터를 이용하여 버그 보고서를 분류하는 일 예를 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0009] 실시 예들에서 사용되는 용어는 가능한 현재 널리 사용되는 일반적인 용어들을 선택하였으나, 이는 당 분야에 종사하는 기술자의 의도 또는 판례, 새로운 기술의 출현 등에 따라 달라질 수 있다. 또한, 특정한 경우는 출원인이 임의로 선정한 용어도 있으며, 이 경우 해당되는 설명 부분에서 상세히 그 의미를 기재할 것이다. 따라서 명세서에서 사용되는 용어는 단순한 용어의 명칭이 아닌, 그 용어가 가지는 의미와 명세서의 전반에 걸친 내용을 토대로 정의되어야 한다.

[0010] 명세서 전체에서 어떤 부분이 어떤 구성요소를 “포함”한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있음을 의미한다. 또한, 명세서에 기재된 “~ 유닛”, “~ 모듈” 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하며, 이는 하드웨어 또는 소프트웨어로 구현되거나 하드웨어와 소프트웨어의 결합으로 구현될 수 있다.

[0011] 또한, 명세서에서 사용되는 “제 1” 또는 “제 2” 등과 같이 서수를 포함하는 용어는 다양한 구성 요소들을 설명하는데 사용할 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로 사용될 수 있다.

[0012] 아래에서는 첨부한 도면을 참고하여 실시 예에 대하여 상세히 설명한다. 그러나 실시 예는 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 예에 한정되지 않는다.

[0013] 도 1은 일 실시예에 따른 버그 보고서를 분류하는 장치의 일 예를 설명하기 위한 도면이다.

[0014] 도 1을 참조하면, 일 실시예에 따른 버그 보고서를 분류하는 장치(100)는 이미지 데이터를 처리하는 CNN 기반의 이미지 채널(110), 텍스트 데이터를 처리하는 CNN 기반의 텍스트 채널(120) 및 코드 데이터를 처리하는 CNN 기반의 코드 채널(130)을 포함한다. 그리고 장치는 각각의 채널(110, 120, 130)에서 특징 벡터를 추출하고, 이러한 특징 벡터들을 각각의 완전 연결 레이어(111, 121, 131)에 입력한다.

[0015] 이후 장치(100)는, 각각의 완전 연결 레이어(111, 121, 131)에서 출력된 벡터들을 요소별 곱셈을 통해 융합한다. 이후 장치는 융합된 벡터를 최종 완전 연결 레이어(140) 및 소프트맥스 레이어(150)에 입력하여 소프트맥스 연산을 수행하고, 버그 보고서를 버그(Bug) 또는 특징(Feature)으로 분류하는 분류 결과를 획득한다.

[0016] 여기서 버그 보고서는, 소프트웨어가 예상하지 못한 잘못된 결과를 내거나, 오류가 발생하거나, 착오가 오작동이 발생하는 등의 문제가 발생하거나, 사용자가 문제가 발생한 것으로 판단했을 때, 문제와 관련된 내용을 기록하는 보고서일 수 있다. 버그 보고서는 소프트웨어 제품을 사용하는 사용자가 작성하여 보고한 것일 수 있고, 소프트웨어 개발자가 작성한 것일 수 있다. 버그 보고서에는 텍스트뿐 아니라, 이미지 및 코드를 포함한 다양한 유형의 정보가 포함될 수 있다. 일 실시예에 따른 버그 보고서의 예는 도 4를 참조하여 후술한다.

[0017] 통상적으로, 버그 보고서를 자동으로 분류함에 있어서, 버그 보고서 내의 텍스트 데이터만을 활용하는 것이 일반적이다. 그러나, 실제 버그 보고서에는 텍스트, 이미지 및 코드를 포함한 다양한 모달리티의 데이터가 존재하며, 텍스트 데이터만을 기반으로 버그 보고서를 분류하는 경우, 이러한 정보들을 활용하지 못하여 분류 성능에 한계가 있다.

[0018] 일 실시예에 따른 장치(100)는 멀티모달 딥러닝을 통해 버그 보고서에 포함된 텍스트, 이미지 및 코드 정보를 결합함으로써 분류 작업을 고도화할 수 있다. 또한, 이를 통해 버그 보고서의 표현을 정교화할 수 있기 때문에 버그 보고서의 분류 성능을 개선할 수 있다.

[0019] 이하 도 2 내지 도 11을 참조하여 도 1에 도시된 장치(100)를 구체적으로 설명한다.

[0020] 도 2는 일 실시예에 따른 버그 보고서를 분류하는 장치의 일 예를 도시한 구성도이다.

[0021] 도 2를 참조하면, 버그 보고서를 분류하는 장치(200)는 프로세서(210) 및 메모리(220)를 포함한다. 설명의 편의를 위하여, 도 2에는 본 발명과 관련된 구성요소들만이 도시되어 있다.

[0022] 따라서, 도 2에 도시된 구성요소들 외에 다른 범용적인 구성요소들이 장치(200)에 더 포함될 수 있다. 또한, 도 2에 도시된 프로세서(210) 및 메모리(220)는 독립된 장치로 구현될 수도 있음은 본 발명과 관련된 기술 분야에서의 통상의 지식을 가진 자에게 자명하다.

[0023] 프로세서(210)는 기본적인 산술, 로직 및 입출력 연산을 수행함으로써, 컴퓨터 프로그램의 명령을 처리할 수 있다. 여기에서, 명령은 메모리(220) 또는 외부 장치(예를 들어, 서버 등)로부터 제공될 수 있다. 또한, 프로세서(210)는 장치(200)에 포함된 다른 구성요소들의 동작을 전반적으로 제어할 수 있다.

[0024] 프로세서(210)는 버그 보고서에서 텍스트 데이터, 코드 데이터 및 이미지 데이터를 추출하고, 각 유형의 데이터로부터 각각 특징 벡터를 추출할 수 있다. 그리고 프로세서(210)는 각각의 특징 벡터들을 융합(fusing)할 수 있다. 이후 프로세서(210)는 결합된 벡터에 대해 소프트맥스(Softmax) 연산을 수행하여 버그 보고서의 카테고리를 분류할 수 있다.

[0025] 프로세서(210)는 다수의 논리 게이트들의 어레이로 구현될 수도 있고, 범용적인 마이크로 프로세서와 이 마이크로 프로세서에서 실행될 수 있는 프로그램이 저장된 메모리의 조합으로 구현될 수도 있다. 예를 들어, 프로세서(210)는 범용 프로세서, 중앙 처리 장치(CPU), 마이크로프로세서, 디지털 신호 프로세서(DSP), 제어기, 마이크로제어기, 상태 머신 등을 포함할 수 한다. 일부 환경에서, 프로세서(210)는 주문형 반도체(ASIC), 프로그램 가능 로직 디바이스(PLD), 필드 프로그램 가능 게이트 어레이(FPGA) 등을 포함할 수도 있다. 예를 들어, 프로세서(210)는 디지털 신호 프로세서(DSP)와 마이크로프로세서의 조합, 복수의 마이크로프로세서들의 조합, 디지털 신호 프로세서(DSP) 코어와 결합된 하나 이상의 마이크로프로세서들의 조합, 또는 임의의 다른 그러한 구성들의 조합과 같은 처리 디바이스들의 조합을 지칭할 수도 있다.

[0026] 메모리(220)는 비-일시적인 임의의 컴퓨터 판독 가능한 기록매체를 포함할 수 있다. 일 예로서, 메모리(220)는 RAM(random access memory), ROM(read only memory), 디스크 드라이브, SSD(solid state drive), 플래시 메모리(flash memory) 등과 같은 비소멸성 대용량 저장 장치(permanent mass storage device)를 포함할 수 있다. 다른 예로서, ROM, SSD, 플래시 메모리, 디스크 드라이브 등과 같은 비소멸성 대용량 저장 장치는 메모리와는 구분되는 별도의 영구 저장 장치일 수 있다. 또한, 메모리(220)에는 운영체제(OS)와 적어도 하나의 프로그램 코드(예를 들어, 도 5 내지 도 11을 참조하여 후술할 동작을 프로세서(210)가 수행하기 위한 코드)가 저장될 수 있다.

[0027] 이러한 소프트웨어 구성요소들은 메모리(220)와는 별도의 컴퓨터에서 판독 가능한 기록매체로부터 로딩될 수 있다. 이러한 별도의 컴퓨터에서 판독 가능한 기록매체는 장치(200)에 직접 연결될 수 있는 기록 매체일 수 있고, 예를 들어, 플로피 드라이브, 디스크, 테이프, DVD/CD-ROM 드라이브, 메모리 카드 등의 컴퓨터에서 판독 가능한 기록매체를 포함할 수 있다.

[0028] 또는, 소프트웨어 구성요소들은 컴퓨터에서 판독 가능한 기록매체가 아닌 통신 모듈을 통해 메모리(220)에 로딩될 수도 있다. 예를 들어, 적어도 하나의 프로그램은 개발자들 또는 어플리케이션의 설치 파일을 배포하는 파일 배포 시스템이 통신 모듈을 통해 제공하는 파일들에 의해 설치되는 컴퓨터 프로그램(예를 들어, 도 5 내지 도 11을 참조하여 후술할 동작을 프로세서(210)가 수행하기 위한 컴퓨터 프로그램 등)에 기반하여 메모리(220)에 로딩될 수 있다.

[0029] 도 3은 일 실시예에 따른 버그 보고서를 분류하는 방법의 일 예를 설명하기 위한 흐름도이다.

[0030] 도 3를 참조하면, 버그 보고서를 분류하는 방법은 도 1 및 도 2에 도시된 장치(100, 200) 또는 프로세서(210)에서 시계열적으로 처리되는 단계들로 구성된다. 따라서, 이하에서 생략된 내용이라고 하더라도 도 1 및 도 2에 도시된 장치(100, 200) 또는 프로세서(210)에 관하여 이상에서 기술된 내용은 도 3의 버그 보고서를 분류하는 방법에도 적용될 수 있다.

[0031] 310 단계에서, 프로세서(210)는 버그 보고서에서 추출된 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성한다.

[0032] 프로세서(210)는 텍스트 데이터에 대한 전처리를 수행할 수 있다. 그리고, 전처리된 텍스트 데이터를 제1 모달 모델을 통해 처리함으로써 제1 벡터를 생성할 수 있다. 예를 들어, 텍스트 데이터에 대한 전처리는 텍스트 데이

터의 대소문자를 통일하고, 대소문자 통일된 텍스트 데이터를 분할하여 토큰화하고, 토큰화된 텍스트 데이터에서 불용어를 제거하는 것일 수 있다.

[0033] 예를 들어 프로세서(210)는, 제1 모달 모델에 포함된 적어도 하나의 커널을 이용하여, 전처리된 텍스트 데이터에 대한 합성곱 연산을 수행하여, 적어도 하나의 합성곱 특징 벡터를 추출할 수 있다. 그리고 프로세서(210)는 적어도 하나의 합성곱 특징 벡터에 대해, 맥스 풀링(max pooling) 연산을 수행하고, 맥스 풀링 연산의 결과들을 결합(concatenate)하여 단일 벡터를 생성할 수 있다. 프로세서(210)는 생성된 단일 벡터를 제1 완결 연결 레이어(Fully Connected layer, FC)에 입력하여 단일 벡터에 ReLU 연산 및 선형 연산을 수행하여 제1 벡터를 생성할 수 있다.

[0034] 이하, 도 4를 참조하여, 버그 보고서의 일 예를 설명하고, 도 5 및 도 6을 참조하여, 프로세서(210)가 제1 벡터를 생성하는 일 예를 설명한다.

[0035] 도 4는 일 실시예에 따른 버그 보고서의 일 예를 나타내는 도면이다.

[0036] 도 4를 참조하면, 버그 보고서에는 텍스트 데이터(410), 코드 데이터(420) 및 이미지 데이터(430)가 포함되어 있다. 일 실시예에 따른 버그 보고서를 분류하는 장치는 텍스트 데이터(410), 코드 데이터(420) 및 이미지 데이터(430)를 각 유형의 데이터를 처리하는 서로 다른 CNN 기반의 모델을 통해 처리하여 특징 벡터를 획득하고, 세 종류의 특징 벡터를 융합하여 버그 보고서의 카테고리를 분류할 수 있다.

[0037] 도 5는 일 실시예에 따른 프로세서가 텍스트 데이터에 대한 전처리를 수행하는 일 예를 설명하기 위한 도면이다.

[0038] 도 5를 참조하면, 프로세서(210)는 버그 보고서에서 추출된 텍스트 데이터(50)에 대소문자를 통일하는 케이스 폴딩(case folding)을 수행할 수 있다. 대소문자 통일은 텍스트 데이터의 대문자와 소문자를 구분하지 않도록 처리하는 것으로, 텍스트 데이터의 일관성을 유지하고, 대소문자로 인하여 발생할 수 있는 혼란을 방지하기 위해 수행된다. 프로세서(210)는 텍스트 데이터(50)에 포함된 대문자(511,512,513,514,515)를 소문자(511',512',513',514',515')로 바꾸어 대소문자의 구분이 없는 텍스트 데이터(51)를 획득할 수 있다.

[0039] 이후 프로세서(210)는 대소문자가 통일된 텍스트 데이터(51)를 구문 단위로 분할하여 토큰화(Tokenization)할 수 있다. 여기서 토큰화는, 텍스트 데이터를 작은 단위로 분할하는 과정을 의미한다. 프로세서(210)는 텍스트 데이터를 문장이나 문단 단위로 분할할 수도 있고, 단어 단위로 분할할 수 있으며, 그보다 더 작은 단위로 분할할 수 있다. 토큰화를 통해 분할된 텍스트 데이터의 개별요소들을 토큰이라고 한다.

[0040] 프로세서(210)는 토큰화된 텍스트 데이터(52)에서 불용어(stop word)를 제거할 수 있다. 불용어는 일반적으로 문맥이나 의미를 파악하는 데 도움이 되지 않거나 중요한 정보를 제공하지 않는 단어나 용어를 의미한다. 프로세서(210)는 메모리(220)에 저장된 불용어 목록과 토큰(520, 531, 532, 533)을 비교하여, 불용어로 판단되는 토큰(531, 532)을 제거할 수 있다. 불용어 목록에는 "the," "and," "is," "in", "it", "for", "to", "of", “on”, “after” 등이 포함될 수 있다. 또한, 프로세서(210)는 불용어를 제거함으로써, 텍스트 데이터의 크기를 줄이고, 텍스트 데이터에서 의미 있는 토큰만 유지할 수 있다. 프로세서(210)는 불용어를 제거함으로써 텍스트 데이터에 대한 연산 및 분석의 효율성을 향상시킬 수 있다. 일 실시예에 따르면, 불용어 목록은 파이썬의 NLTK(Natural Language Toolkit) 라이브러리의 불용어 목록과 같거나 유사할 수 있다. 또한 프로세서(210)는 숫자로 된 토큰(533)을 제거할 수 있다.

[0041] 도 6은 일 실시예에 따른 프로세서가 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성하는 일 예를 설명하기 위한 도면이다.

[0042] 도 6을 참조하면, 프로세서(210)는 전처리된 텍스트 데이터를 제1 모달 모델에 입력할 수 있다. 제1 모달 모델은 CNN 기반의 모델일 수 있다. 제1 모달 모델은 합성곱 레이어(610), 맥스 풀링 레이어(620), 결합 레이어(630, Concat layer) 및 제1 완전 연결 레이어(640)를 포함할 수 있다.

[0043] 일 실시예에 따르면, 프로세서(210)는 불용어가 제거된 텍스트 데이터를 벡터화하여, 벡터화된 텍스트 데이터를 합성곱 레이어(610)에 입력할 수 있다. 텍스트 데이터의 벡터화는 제1 모달 모델의 임베딩 레이어(Embedding layer)에서 수행될 수 있으며, Word2Vec, GloVe, FastText와 같은 사전 훈련된 모델을 사용하여 단어 임베딩(word embeddings)을 생성하는 것일 수 있으나, 이에 한정되지 않으며, 다른 텍스트 토큰의 벡터화 방법이 수행될 수 있다.

[0044] 일 실시예에 따르면, 합성곱 레이어(610)는 각각 2, 3, 4, 5의 벡터 사이즈의 4개의 커널로 구성될 수 있으나,

이에 한정되지 않는다. 합성곱 레이어(610)의 출력 채널은 64개일 수 있으나, 이에 한정되지 않는다. 프로세서(210)는 벡터화된 텍스트 데이터를 합성곱 레이어(610)에 입력하여 컨볼루션 연산을 수행한다. 그리고 컨볼루션 연산된 결과를 맥스 풀링 레이어(620)에 입력하여 맥스 풀링 연산을 수행한다. 그리고 프로세서(210)는 맥스 풀링 연산 결과인 벡터들을 결합 레이어(630)에서 결합하여 단일 벡터를 생성한다. 결합 레이어(Concat layer)는 여러 개의 벡터를 결합하여 하나의 벡터를 생성하는 연산을 수행한다. 이후 프로세서(210)는 생성된 단일 벡터를 제1 완전 연결 레이어(640)에 입력하고, 제1 완전 연결 레이어(640)에서 ReLU 연산 및 선형 연산을 수행하여 제1 벡터를 생성한다.

[0045] 다시 도 3을 참조하면, 320 단계에서, 프로세서(210)는 버그 보고서에서 추출된 코드 데이터를 제2 모달(modal) 모델을 통해 처리함으로써 제2 벡터를 생성한다.

[0046] 프로세서(210)는 코드 데이터에 대한 전처리를 수행할 수 있다. 그리고, 전처리된 코드 데이터를 제2 모달 모델을 통해 처리함으로써 제2 벡터를 생성할 수 있다. 예를 들어, 코드 데이터에 대한 전처리는 코드 데이터에 포함된 자연어를 제거하고, 자연어가 제거된 코드 데이터에서 코드의 구조를 나타내는 부분을 이스케이프 시퀀스(escape sequence)로 대체하고, 대체된 코드 데이터의 대소문자를 통일하고, 대소문자 통일된 코드 데이터를 토큰화하는 것일 수 있다. 예를 들어 프로세서(210)는, 제2 모달 모델에 포함된 적어도 하나의 커널을 이용하여, 전처리된 텍스트 데이터에 대한 합성곱 연산을 수행하여, 적어도 하나의 합성곱 특징 벡터를 추출할 수 있다. 그리고 프로세서(210)는 적어도 하나의 합성곱 특징 벡터에 대해, 맥스 풀링(max pooling) 연산을 수행하고, 맥스 풀링 연산의 결과들을 결합(concatenate)하여 단일 벡터를 생성할 수 있다. 프로세서(210)는 생성된 단일 벡터를 제2 완결 연결 레이어(Fully Connected layer)에 입력하여 단일 벡터에 ReLU 연산 및 선형 연산을 수행하여 제2 벡터를 생성할 수 있다.

[0047] 이하, 도 7을 참조하여, 프로세서(210)가 코드 데이터에 대한 전처리를 수행하는 일 예를 설명한다.

[0048] 도 7은 일 실시예에 따른 프로세서가 코드 데이터에 대한 전처리를 수행하는 일 예를 설명하기 위한 도면이다.

[0049] 프로세서(210)는 코드 데이터(70)에서 자연어(711, 712)를 제거할 수 있다. 자연어는 코드 데이터에 포함된 주석(Comments)을 의미할 수 있다. 예를 들어, 자연어는 코드 작성자가 코드의 가독성을 높이고 다른 개발자들에게 의도나 동작을 설명하기 위해 메모 형태로 포함한 텍스트일 수 있다.

[0050] 프로세서(210)는 자연어 부분이 제거된 코드 데이터(71)에서 코드의 구조를 나타내는 부분을 이스케이프 시퀀스(escape sequence)로 대체하는 스페셜 토큰 대체(special token replacement)를 수행할 수 있다. 여기서 코드의 구조를 나타내는 부분이란, 코드에서의 들여쓰기 또는 줄바꿈을 의미하는 것일 수 있다. 여기서 이스케이프 시퀀스(escape sequence)란, 역슬래시와 문자의 조합으로, 특정한 제어 문자나 특수문자를 나타내는 문자 조합이다. 프로세서(210)는 코드 데이터에 포함된 '들여쓰기' 및 '줄바꿈'을 각각 '\n'(721) 및 '\t'(722)로 대체할 수 있다. 프로세서(210)는 코드의 구조를 나타내는 부분을 이스케이프 시퀀스(721, 722)로 대체함으로써 코드의 구조와 형식을 보존하고 코드를 효과적으로 분석할 수 있다.

[0051] 이후 프로세서(210)는 스페셜 토큰 대체가 수행된 코드 데이터(72)의 대소문자를 통일할 수 있다. 예를 들어, 프로세서(210)는 코드 데이터에 포함된 대문자(731)를 소문자(732)로 변경하여, 대소문자의 구분이 없는 코드 데이터(73)를 획득할 수 있다. 그리고 프로세서(210)는 대소문자가 통일된 코드 데이터(73)를 분할하여 토큰화된 코드 데이터(74)를 획득할 수 있다.

[0052] 제2 모달 모델은 합성곱 레이어, 맥스 풀링 레이어, 결합 레이어(Concat layer) 및 제2 완전 연결 레이어를 포함할 수 있다. 합성곱 레이어의 커널은 4개일 수 있으나 이에 한정되지 않는다. 커널의 크기는 2,3,4,5 벡터 사이즈일 수 있으나, 이에 한정되지 않는다. 출력 채널은 64개일 수 있으나, 이에 한정되지 않는다. 제2 모달 모델은 제1 모달 모델과 동일한 모델일 수 있고, 각 레이어의 개수 또는 커널의 크기가 다른 모델일 수 있다.

[0053] 다시 도 3을 참조하면, 330 단계에서, 프로세서(210)는 버그 보고서에서 추출된 이미지 데이터를 제3 모달(modal) 모델을 통해 처리함으로써 제3 벡터를 생성한다.

[0054] 프로세서(210)는 이미지 데이터에 대한 전처리를 수행할 수 있다. 그리고, 전처리된 이미지 데이터를 제3 모달 모델을 통해 처리함으로써 제3 벡터를 생성할 수 있다. 예를 들어, 이미지 데이터에 대한 전처리는 이미지 데이터에 포함된 모든 이미지의 크기를 동일한 크기로 크기 조절하고, 크기 조절된 이미지 데이터의 분포 및 스케일을 정규화하는 것일 수 있다. 예를 들어 프로세서(210)는, 제3 모달 모델의 적어도 하나의 합성곱 레이어에 전처리된 이미지 데이터를 입력할 수 있다. 또한 프로세서(210)는 전처리된 이미지 데이터에 합성곱 연산 및 ReLU 연산을 수행하여, 합성곱 특징 벡터를 추출할 수 있다. 그리고, 프로세서(210)는 상기 합성곱 특징 벡터에

대해, 맥스 풀링(max pooling) 연산을 수행하고, 맥스 풀링(max pooling) 연산의 결과를 완결 연결 레이어(Fully Connected layer)에 입력하여 제3 벡터를 추출할 수 있다.

[0055] 이하, 도 8 및 9를 참조하여, 프로세서(210)가 이미지 데이터를 제3 모달(modal) 모델을 통해 처리함으로써 제3 벡터를 생성하는 예를 설명한다.

[0056] 도 8은 일 실시예에 따른 프로세서가 이미지 데이터에 대한 전처리를 수행하는 일 예를 설명하기 위한 도면이다.

[0057] 프로세서(210)는 이미지 데이터에 포함된 모든 이미지(80)의 크기를 동일한 크기로 크기 조절(Resize)할 수 있다. 프로세서(210)는 이미지(80)를 258 × 258의 크기로 조절할 수 있으나, 이미지의 크기는 이에 한정되지 않는다. 프로세서(210)는 Torchvision의 변환 모듈(transforms module)을 사용하여 크기 조절을 수행할 수 있다. 그리고, 프로세서(210)는 크기 조절된 이미지 데이터(81)의 분포 및 스케일을 정규화(Normalize)할 수 있다. 프로세서(210)는 이미지의 픽셀 값을 정규화하기 위해 Torchvision의 변환 모듈(transforms module)을 사용할 수 있으며, 이를 통해 픽셀 값은 -1에서 1 범위로 정규화될 수 있다. 프로세서(210)는 크기 조절 및 정규화된 이미지 데이터(82)를 제3 모달의 입력 데이터로 사용할 수 있다.

[0058] 도 9는 일 실시예에 따른 프로세서가 이미지 데이터를 제3 모달(modal) 모델을 통해 처리함으로써 제3 벡터를 생성하는 일 예를 설명하기 위한 도면이다.

[0059] 프로세서(210)는 전처리된 이미지 데이터를 제3 모달 모델에 입력한다. 전 처리된 이미지 데이터는 벡터화된 데이터일 수 있다. 제3 모달 모델은 합성곱 레이어(910), 맥스 풀링 레이어(920) 및 제3 완전 연결 레이어(930)를 포함할 수 있다. 프로세서(210)는 합성곱 레이어에서 특정 크기의 커널(940)로 전처리된 이미지 데이터(950)를 스캔하면서 합성곱 연산을 수행한다. 합성곱 레이어(910)의 커널(940)의 크기는 5 × 5일 수 있으나, 이에 한정되지 않는다. 프로세서(210)는 ReLU 활성화 함수를 이용하여 합성곱 연산 결과에 비선형성을 부여하고, 특징을 강조할 수 있다. 도 9에는 합성곱 레이어(910)가 하나의 레이어로 표현되었으나, 합성곱 레이어(910)는 2 이상의 레이어일 수 있다. 프로세서는 합성곱 레이어(910)의 출력을 맥스 풀링 레이어(920)에 입력하여 이미지의 크기를 줄일 수 있다. 도 9에는 맥스 풀링 레이어(920)가 하나의 레이어로 표현되었으나, 맥스 풀링 레이어(920)는 2 이상의 레이어일 수 있고, 프로세서(210)는 합성곱 레이어(910)의 출력에 대해 맥스 풀링 연산을 2 이상 수행할 수 있다. 그리고, 프로세서(210)는 맥스 풀링 레이어(920)의 출력을 제3 완전 연결 레이어(930)에 입력하고, 완전 연결 레이어(930)의 출력으로 제3 벡터를 획득할 수 있다. 제3 벡터는 1 × 1 × 256 크기의 벡터일 수 있으나, 이는 예시에 불과하며, 제3 모달 모델의 각 레이어의 개수와 크기 등 구체적인 아키텍처에 따라 달라질 수 있다.

[0060] 다시 도 3을 참조하면, 단계 340에서, 프로세서(210)는 제1 벡터, 제2 벡터 및 제3 벡터를 이용하여 버그 보고서를 분류한다.

[0061] 프로세서(210)는 텍스트 데이터를 처리하는 제1 모달 모델로부터 획득한 제1 벡터, 코드 데이터를 처리하는 제2 모달 모델로부터 획득한 제2 벡터 및 이미지 데이터를 처리하는 제3 모달 모델로부터 획득한 제3 벡터를 하나의 벡터로 융합한 제4 벡터를 생성하고, 제4 벡터를 이용하여 버그 보고서를 분류할 수 있다. 일 실시예에 따른 프로세서(210)는 제1 벡터, 제2 벡터 및 제3 벡터를 융합하여 제4 벡터를 획득하기 위해, 제1 벡터, 제2 벡터 및 제3 벡터에 요소별 곱셈(element-wise multiplication)을 수행할 수 있다. 다른 실시예에 따른 프로세서(210)는 가중합(weighted sum) 또는 결합(concatenation) 연산을 수행하여 제4 벡터를 획득할 수 있다.

[0062] 이하, 도 10 및 11을 참조하여, 프로세서가 제1 벡터, 제2 벡터 및 제3 벡터를 이용하여 버그 보고서를 분류하는 예를 설명한다.

[0063] 도 10은 일 실시예에 따른 프로세서가 제1 벡터, 제2 벡터 및 제3 벡터를 이용하여 버그 보고서를 분류하는 일 예를 설명하기 위한 흐름도이다.

[0064] 단계 1010을 참조하면, 일 실시예에 따른 프로세서(210)는 제1 벡터, 제2 벡터 및 제3 벡터에 요소별 곱셈(element-wise multiplication)을 수행하여, 제4 벡터를 획득할 수 있다. 여기서 요소별 곱셈은 크기가 같은 복수의 벡터에서 동일한 위치에 있는 원소들끼리 곱한 결과를 산출하는 것을 의미한다.

[0065] 단계 1020을 참조하면, 일 실시예에 따른 프로세서(210)는 제4 벡터를 이용하여 버그 보고서를 분류할 수 있다.

[0066] 도 11은 일 실시예에 따른 프로세서가 제4 벡터를 이용하여 버그 보고서를 분류하는 일 예를 설명하기 위한 도

면이다.

[0067] 프로세서(210)는 제1 모달 모델(1140), 제2 모달 모델(1150) 및 제3 모달 모델(1160)의 출력인 제1 벡터, 제2 벡터 및 제3 벡터에 대해 요소별 곱셈을 수행하여, 제4 벡터를 생성하고, 텍스트 데이터, 코드 데이터 및 이미지 데이터의 특징 벡터를 공통 공간에 투영한다. 이후 프로세서(210)는 제4 벡터를 최종 완전 연결 레이어(1170)에 입력하고, 최종 완전 연결 레이어(1170)로부터 출력을 획득하고, 상기 출력을 소프트맥스 레이어(1180)에 입력하여 소프트맥스 레이어(1180)로부터 버그 보고서가 '버그' 카테고리 또는 '특징' 카테고리에 해당할 확률로 표현되는 버그 보고서의 분류 결과를 획득할 수 있다. 소프트맥스 레이어(1180)의 출력은 [0,1] 범위에서 정규화된 클래스별 확률이다. 이때, 본 발명의 멀티 모달 모델은 손실 함수로 교차 엔트로피 손실 함수(Cross Entropy Loss Function)를 사용하여 훈련될 수 있다. 교차 엔트로피 손실 함수는 소프트맥스 레이어(1180)의 출력과 실제 클래스 레이블 간의 차이를 측정하고, 분류의 정확도를 평가하는데 사용될 수 있다.

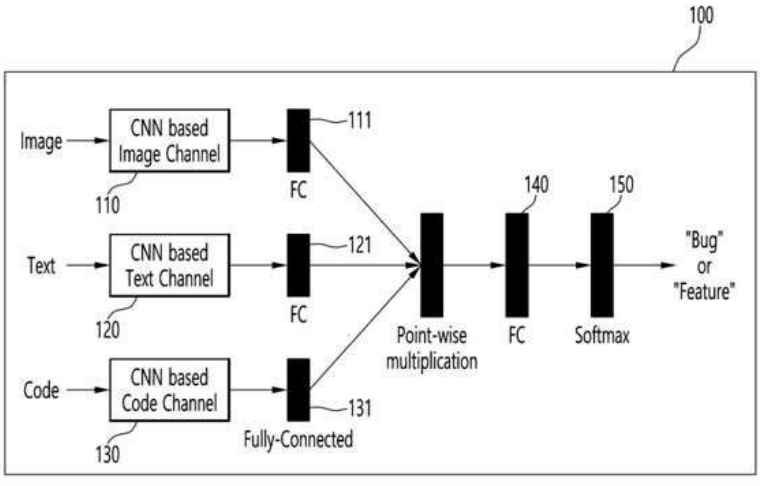
[0068] 여기서, '특징'은 버그가 아닌 것을 의미할 수 있다. 본 명세서에는 버그 보고서를 버그 또는 특징의 카테고리로 분류하는 방법을 예시하였으나, 이에 한정되지 않고, 버그, 개선, 질문의 카테고리로도 분류할 수 있으며, 프로덕션 버그 보고서 및 테스트 버그 보고서로 분류하는 것도 본 발명의 버그 보고서 분류 방법에 의해 구현될 수 있음은 본 발명과 관련된 기술 분야에서의 통상의 지식을 가진 자에게 자명하다.

[0069] 한편, 상술한 방법은 컴퓨터에서 실행될 수 있는 프로그램으로 작성 가능하고, 컴퓨터로 읽을 수 있는 기록매체를 이용하여 상기 프로그램을 동작시키는 범용 디지털 컴퓨터에서 구현될 수 있다. 또한, 상술한 방법에서 사용된 데이터의 구조는 컴퓨터로 읽을 수 있는 기록매체에 여러 수단을 통하여 기록될 수 있다. 상기 컴퓨터로 읽을 수 있는 기록매체는 마그네틱 저장매체(예를 들면, 롬, 램, USB, 플로피 디스크, 하드 디스크 등), 광학적 판독 매체(예를 들면, 시디롬, 디브이디 등)와 같은 저장매체를 포함한다.

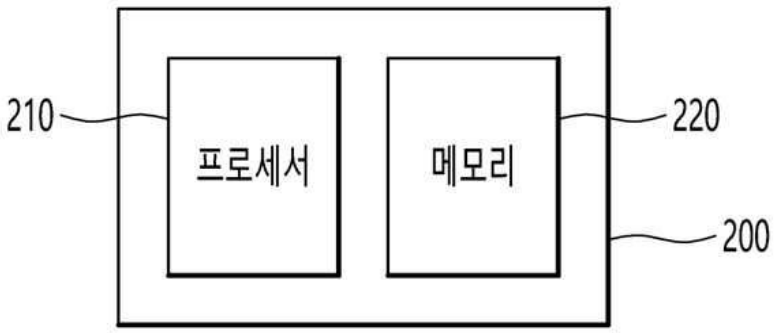
[0070] 본 실시예와 관련된 기술 분야에서 통상의 지식을 가진 자는 상기된 기재의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 방법들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 하며, 권리 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점을 포함하는 것으로 해석되어야 할 것이다.

도면

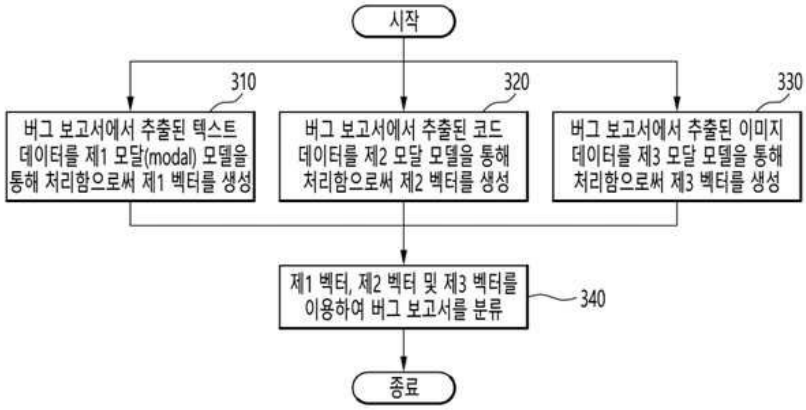
도면1



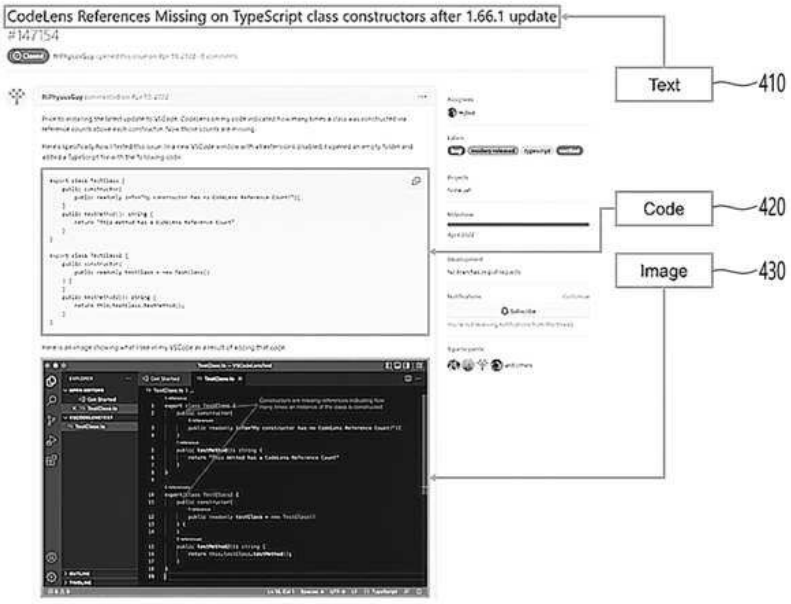
도면2


210
프로세서
메모리
220
200

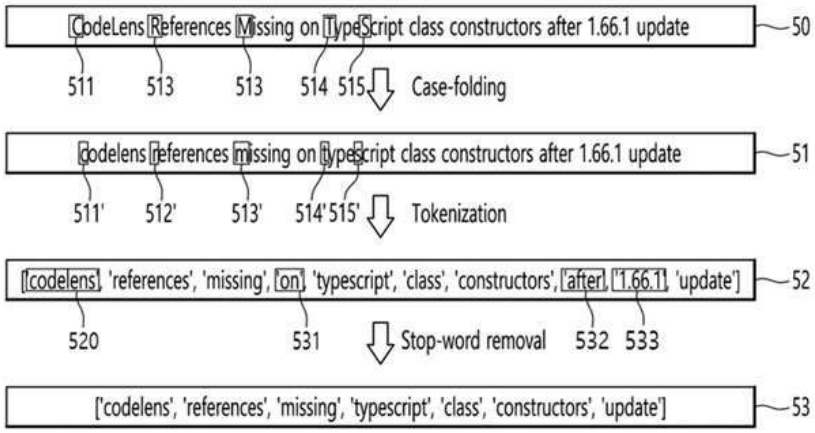
도면3


시작
310
320
330
버그 보고서에서 추출된 텍스트 데이터를 제1 모달(modal) 모델을 통해 처리함으로써 제1 벡터를 생성
버그 보고서에서 추출된 코드 데이터를 제2 모달 모델을 통해 처리함으로써 제2 벡터를 생성
버그 보고서에서 추출된 이미지 데이터를 제3 모달 모델을 통해 처리함으로써 제3 벡터를 생성
제1 벡터, 제2 벡터 및 제3 벡터를 이용하여 버그 보고서를 분류
340
종료

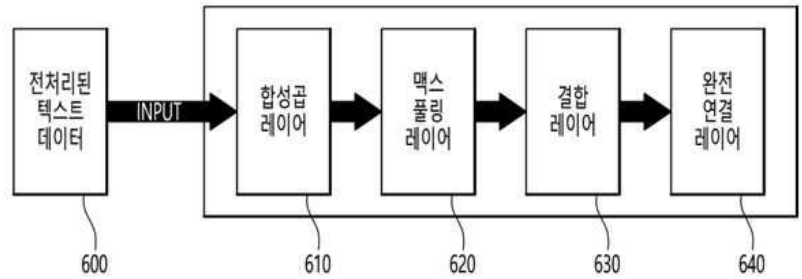
도면4


CodeLens References Missing on TypeScript class constructors after 1.66.1 update
#147154
Text
410
Code
420
Image
430

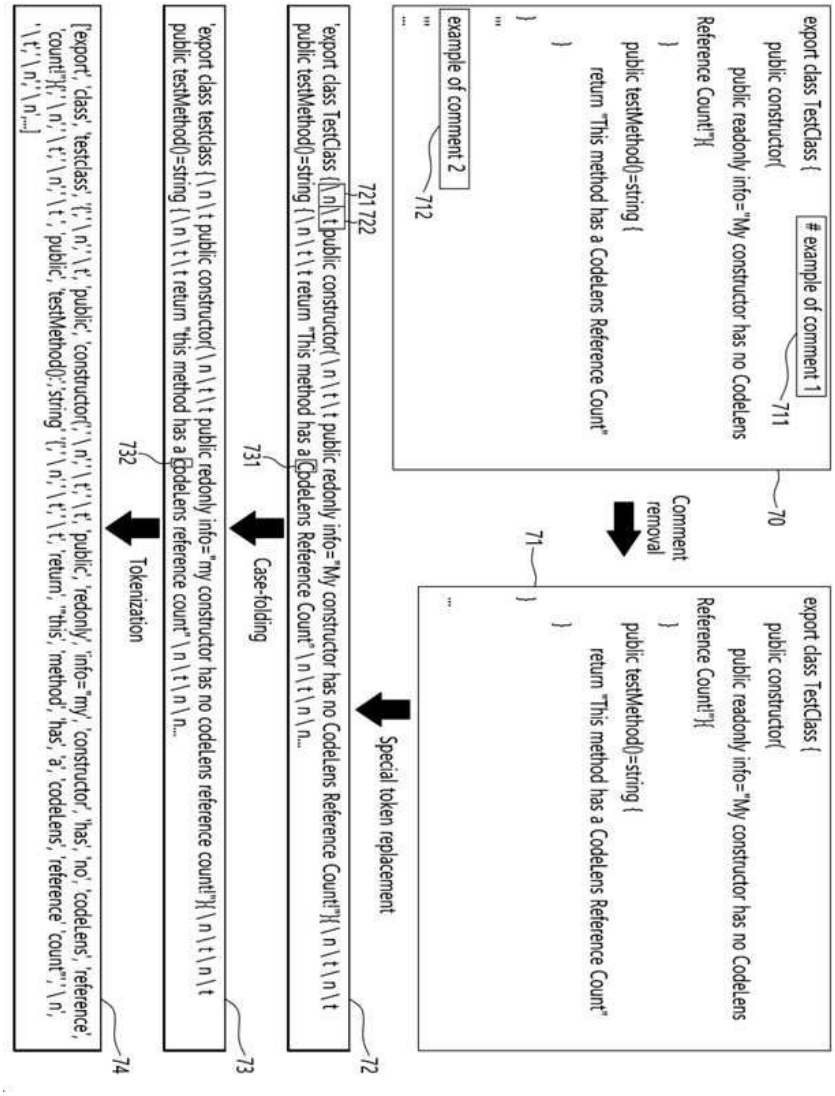
도면5


CodeLens References Missing on TypeScript class constructors after 1.66.1 update
50
511
513
513
514
515
Case-folding
codelens references missing on typescript class constructors after 1.66.1 update
51
511'
512'
513'
514'
515'
Tokenization
['codelens', 'references', 'missing', 'on', 'typescript', 'class', 'constructors', 'after', '1.66.1', 'update']
52
520
531
Stop-word removal
532
533
['codelens', 'references', 'missing', 'typescript', 'class', 'constructors', 'update']
53

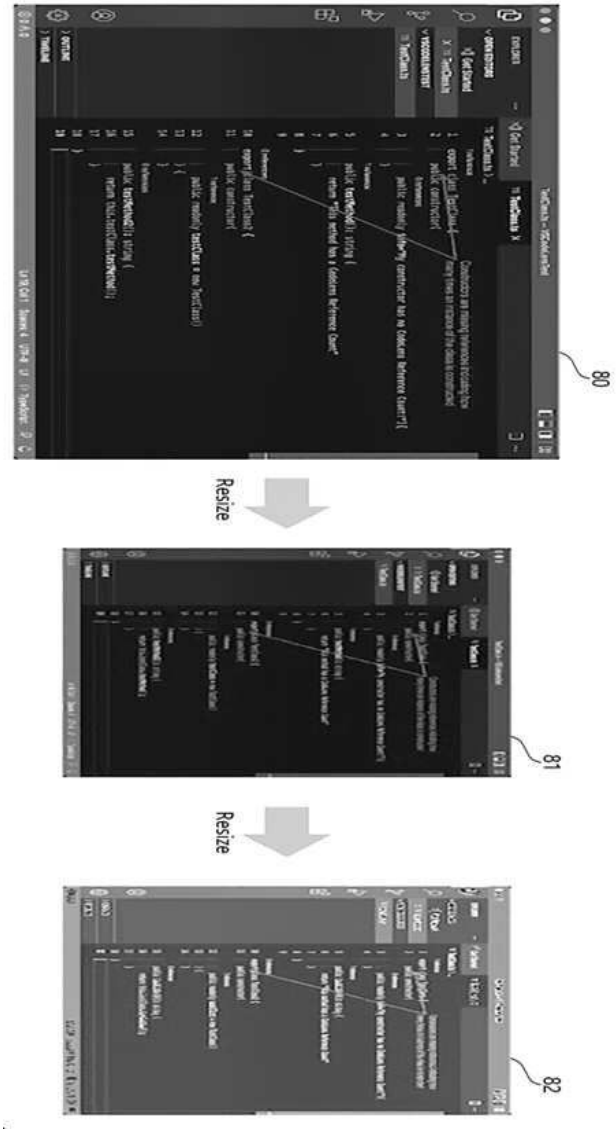
도면6


전처리된 텍스트 데이터
INPUT
합성곱 레이어
맥스 풀링 레이어
결합 레이어
완전 연결 레이어
600
610
620
630
640

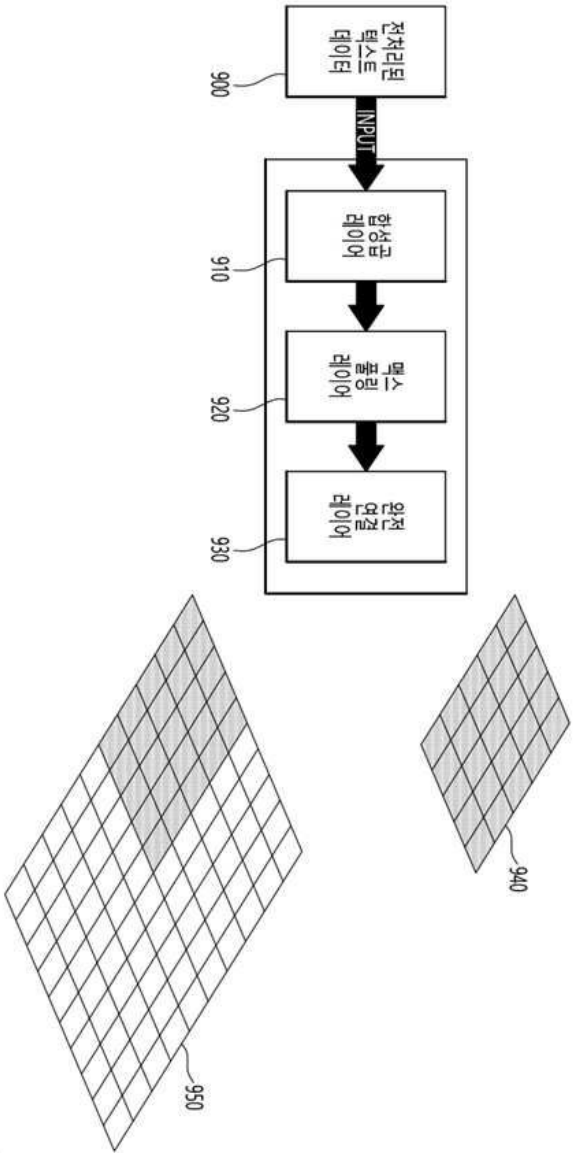
도면7


export class TestClass {
public constructor(
public readonly info="My constructor has no CodeLens Reference Count!"){
}
public testMethod()=string {
return "This method has a CodeLens Reference Count"
}
}
...
example of comment 1
example of comment 2
...
711
712
70
Comment removal
71
Special token replacement
721 722
72
'export class TestClass {\n\t\t public redonly info="My constructor has no CodeLens Reference Count!"}\n\t\t\n\t
public testMethod()=string {\n\t\t return "This method has a CodeLens Reference Count" \n\t\n\n...
731
Case-folding
73
'export class testclass {\n\t public constructor(\n\t\t\t public redonly info="my constructor has no codeLens reference count!"}\n\t\t\n\t
public testMethod()=string {\n\t\t return "this method has a codeLens reference count" \n\t\n\n...
732
Tokenization
74
['export', 'class', 'testclass', '{', '\n', '\t', 'public', 'constructor(', '\n', '\t', '\t', 'public', 'redonly', 'info=', 'my', 'constructor', 'has', 'no', 'codeLens', 'reference',
'count!"}', '\n', '\t', '\n', '\t', 'public', 'testMethod()', 'string' '{', '\n', '\t', '\t', 'return', "this", 'method', 'has', 'a', 'codeLens', 'reference' 'count"', '\n',
'\t', '\n', '\n',...]

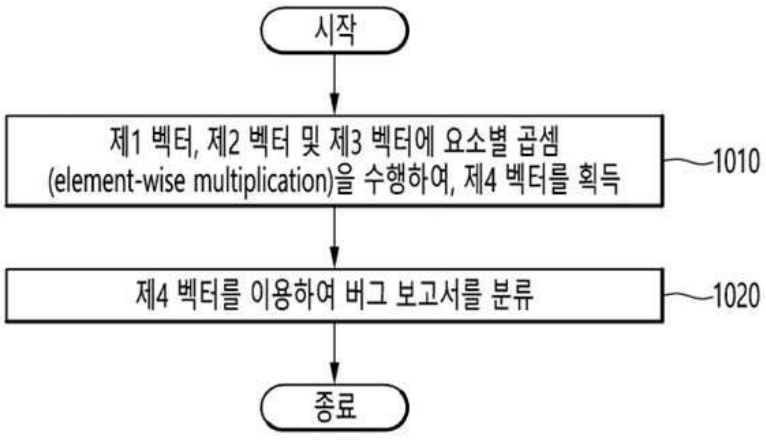
도면8



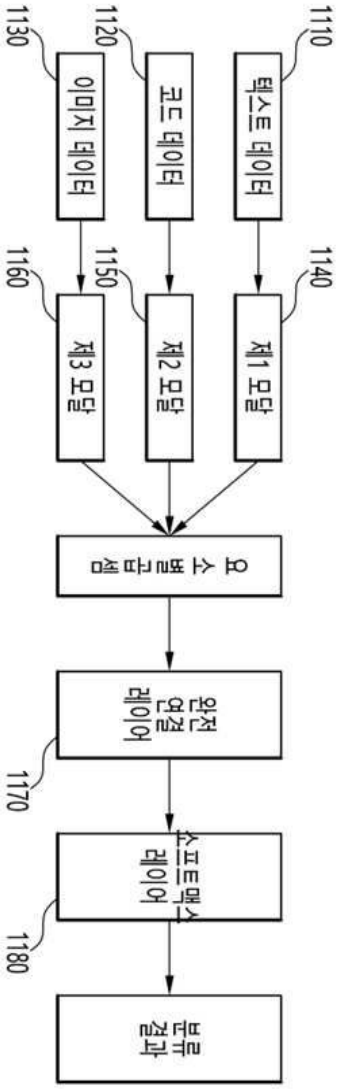
도면9


전처리된 텍스트 데이터
900
INPUT
합성곱 레이어
910
맥스풀 레이어
920
완전연결 레이어
930
940
950

도면10


시작
제1 벡터, 제2 벡터 및 제3 벡터에 요소별 곱셈 (element-wise multiplication)을 수행하여, 제4 벡터를 획득
1010
제4 벡터를 이용하여 버그 보고서를 분류
1020
종료

도면11


1110
텍스트 데이터
1120
코드 데이터
1130
이미지 데이터
1140
제1 모달
1150
제2 모달
1160
제3 모달
융합 모달
1170
완전 연결 레이어
1180
소프트맥스 레이어
출력 결과