

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)(11) 공개번호 10-2024-0070232
(43) 공개일자 2024년05월21일

(51) 국제특허분류(Int. Cl.)
G06F 8/75 (2018.01) G06F 40/279 (2020.01)
G06F 8/41 (2018.01) G06N 20/20 (2019.01)
(52) CPC특허분류
G06F 8/751 (2013.01)
G06F 40/279 (2020.01)
(21) 출원번호 10-2022-0151807
(22) 출원일자 2022년11월14일
심사청구일자 2022년11월14일

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암
동5가)
(72) 발명자
임희석
서울특별시 성북구 안암로 145 자연계캠퍼스 애
기능생활관 311호 NLP&AI연구실
김경민
서울특별시 종로구 평창길 31, B동 203호
(74) 대리인
김동용

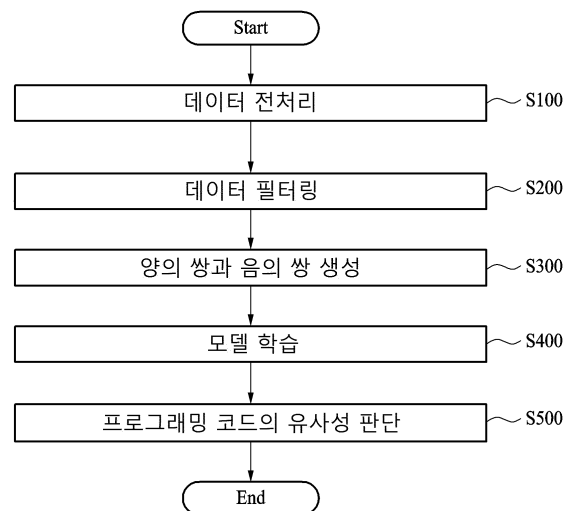
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 교차 검증된 앙상블 및 필터링 전략에 기반한 프로그래밍 코드의 유사성 판단 방법 및 장치

(57) 요약

교차 검증된 앙상블 및 필터링 전략에 기반한 프로그래밍 코드의 유사성 판단 모델 생성 방법이 개시된다. 상기 유사성 판단 모델 생성 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되고, 어느 하나의 언어로 작성된 로 데이터에 대한 전처리를 수행하는 단계, 전처리된 데이터에 대한 필터링을 수행하는 단계, 학습을 위한 양의 쌍들(positive pairs)과 음의 쌍들(negative pairs)을 생성하는 단계, 및 생성된 양의 쌍들과 음의 쌍들을 이용하여 사전학습 언어 모델을 학습시키는 단계를 포함한다.

대표도 - 도3



(52) CPC특허분류

G06F 8/423 (2013.01)

G06F 8/427 (2013.01)

G06N 20/20 (2021.08)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711152697
과제번호	2020-0-00368-003
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	차세대인공지능핵심원천기술개발
연구과제명	뉴럴-심볼릭(neural-symbolic) 모델의 지식 학습 및 추론 기술 개발
기 여 율	1/1
과제수행기관명	고려대학교산학협력단
연구기간	2022.01.01 ~ 2022.12.31

명세서

청구범위

청구항 1

적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 프로그래밍 코드의 유사성 판단 모델 생성 방법에 있어서,

어느 하나의 언어로 작성된 로 데이터(raw data)에 대한 전처리를 수행하는 단계;

전처리된 데이터에 대한 필터링을 수행하는 단계;

학습을 위한 양의 쌍들(positive pairs)과 음의 쌍들(negative pairs)을 생성하는 단계; 및

생성된 양의 쌍들과 음의 쌍들을 이용하여 사전학습 언어 모델을 학습시키는 단계를 포함하는 유사성 판단 모델 생성 방법.

청구항 2

제1항에 있어서,

상기 전처리를 수행하는 단계는,

개행(new line) 제거, 띄어쓰기 제거, 주석 제거, 및 널 스페이스 제거 중 적어도 하나를 수행하는,

유사성 판단 모델 생성 방법.

청구항 3

제1항에 있어서,

상기 필터링을 수행하는 단계는,

상기 전처리된 데이터에 포함된 제1 데이터에 대한 해시 테이블을 생성하고, 상기 해시 테이블을 이용하여 상기 제1 데이터와 상기 전처리된 데이터에 포함된 제2 데이터에서 중복된 데이터를 제거하는 제1 필터링 단계를 포함하는,

유사성 판단 모델 생성 방법.

청구항 4

제3항에 있어서,

상기 필터링을 수행하는 단계는,

제1 필터링 단계에 의해 제1 필터링된 데이터에 대하여, 모든 개행을 이어 붙여 각 행의 앞과 뒤에 존재하는 모든 공백과 문자열 사이의 개행을 소거 후 교집합 값만을 제거하는 제2 필터링 단계를 포함하는,

유사성 판단 모델 생성 방법.

청구항 5

제4항에 있어서,

상기 제2 필터링 단계에 의해 제2 필터링된 데이터에 대하여, 공백을 기준으로 모든 단어 간의 비교를 통해 중복 데이터를 제거하는 제3 필터링 단계를 포함하는,

유사성 판단 모델 생성 방법.

청구항 6

제5항에 있어서,

상기 생성하는 단계는,

BM25 알고리즘 또는 BM25L 알고리즘을 이용하여, 상기 제3 필터링 단계에 의해 제3 필터링된 데이터로부터 상기 양의 쌍들과 상기 음의 쌍들을 생성하는,

유사성 판단 모델 생성 방법.

청구항 7

제6항에 있어서,

상기 학습시키는 단계는 교차 검증 앙상블 기법을 이용하여 상기 사전학습 언어 모델을 학습시키고,

상기 사전학습 언어 모델은 graphcodebert 또는 codebert-mlm인,

유사성 판단 모델 생성 장치.

발명의 설명

기술 분야

[0001] 본 발명은 프로그램 소스 코드에서의 동일한 코드 언어 간 유사성에 대한 비교 및 판단이 가능한 인공 지능(Artificial Intelligence, AI) 언어 모델에 관한 것이다.

배경 기술

[0002] BERT(Bidirectional Encoder Representations from Transformers), GPT(Generative Pre-Training), 및 XLNET(eXtra Long NETwork)와 같은 사전학습된 언어 모델(Pre-Trained Language Models, PLMs)는 최근 몇 년 간 자연어 처리(Natural Language Processing, NLP) 분야에서 기계 독해(Machine Reading Comprehension, MRC), 개체명 인식(Named Entity Recognition, NER), 관계 추출(relation extraction)과 같은 하위 문제(downstream tasks)에서 엄청난 성능 향상을 이끌었다. 이러한 PLMs는 트랜스포머 아키텍처를 기반으로 하며, 먼저 대용량의 비지도 텍스트 코퍼스(unsupervised text corpus)로 모델을 사전학습(pre-train)하고 하위 문제에서 학습 데이터로 미세조정(finetune)을 수행한다. NLP에서의 사전학습과 미세조정 모델 접근은 프로그래밍 언어(programming language)를 위한 사전학습의 발전을 촉진시켰다.

[0003] CodeBERT, graphcodebert는 자연어와 프로그래밍 언어를 모두 학습한 언어모델로, 코드 검색(code search), 코드 문서 생성(code document generation)과 같이 코드에 최적화된 PLM이다. 두 모델 모두 자연어 문서(natural language documents)와 함께 6 개의 프로그래밍 언어의 함수들(functions)을 포함하는 CodeSearchNet 데이터셋으로 사전학습 되었다.

[0004] CodeBERT의 경우 사전학습 과정에서 표준 MLM(standard masked language modeling)과 RTD(replaced token detection)를 포함한 목적 함수(objective function)를 학습하고, graphcodebert는 코드의 의미 구조(semantic structure)를 활용하여 코드 표현(code representation)을 학습하였다. 코드 표현을 학습한 이러한 사전학습 모델들은 프로그래밍 언어 입력 시, 그 유사성(similarity)을 분류할 수 있다는 점에서 코드를 학습하지 않은 일반적인 PLM과 차이가 있고 더 범용적이라 할 수 있다.

[0005] 최근 전 세계적으로 양질의 소프트웨어를 공급할 소프트웨어 개발자와 전문가가 심각하게 부족한 것으로 나타났다. 제한된 소프트웨어 개발자의 공급 하에 소프트웨어 생산성을 증대시키기 위해서는 자동화된 방식을 기반을 한 분석, 개발 및 유지 관리할 수 있는 방법을 미리 갖추는 것이 필수적이고, 세계적으로 많은 연구가 진행되고 있는 분야이다. 이를 위해 첫 번째 단계로, 두 코드 간 동일한 결과물을 산출할 수 있는지의 여부를 판단하는 알고리즘은 생산성을 촉진시키기 위해 매우 중요하다.

[0006] 본 발명에서는 자동화된 방식으로 코드의 유사성을 판별할 수 있는 언어모델, 장치, 및 시스템을 제안한다. 효과적인 언어모델 학습을 위한 3 단계의 전략적인 데이터 필터링 과정은 학습과 테스트의 중복성을 완전히 없앨 수 있으며, 성능이 검증된 Okapi 시스템의 BM25, BM25L 등의 알고리즘으로 양의 쌍(positive pairs)과 음의 쌍(negative pairs)을 생성하였다. 또한, 교차 검증 앙상블(Cross-validated ensemble) 방법은 데이터 불균형이 존재하는 모델 학습 과정에서 정확하고 효과적인 성능을 추출할 수 있다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 대한민국 공개특허 제2020-0080104호 (2020.07.06. 공개)

발명의 내용

해결하려는 과제

[0008] 본 발명이 이루고자 하는 기술적인 과제는 다양한 프로그램 소스 코드에서 동일한 코드 언어 간 유사성을 비교 및 판단할 수 있는 장치 및 그 방법을 제공하는 것이다.

과제의 해결 수단

[0009] 본 발명의 일 실시예에 따른 유사성 판단 모델 생성 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 유사성 판단 모델 생성 방법으로써, 어느 하나의 언어로 작성된 로 데이터에 대한 전처리를 수행하는 단계, 전처리된 데이터에 대한 필터링을 수행하는 단계, 학습을 위한 양의 쌍들(positive pairs)과 음의 쌍들(negative pairs)을 생성하는 단계, 및 생성된 양의 쌍들과 음의 쌍들을 이용하여 사전학습 언어 모델을 학습시키는 단계를 포함한다.

발명의 효과

[0010] 본 발명의 실시예들에 의한 경우, 파이썬(Python), 자바(Java), C++, C, Ruby, C# 등과 같은 다양한 프로그래밍 언어에서 코드의 중복성을 제거하기 위한 유사성을 판단할 수 있다.

[0011] 또한, 토큰 단위가 아닌 문장 단위로 코드 간 유사성을 검사하고, 교차 검증과 데이터 필터링을 통해 높은 성능을 나타낸다.

도면의 간단한 설명

[0012] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1은 본 발명의 전체 개요도를 도시한다.

도 2는 k-폴드 교차 검증을 설명하기 위한 도면이다.

도 3은 본 발명의 일 실시예에 의한 유사성 판단 모델 생성 방법을 설명하기 위한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0013] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0014] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0015] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0016] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어"

있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

- [0017] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0018] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.
- [0019] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.
- [0020] 본 발명에서는 학습 데이터셋의 중복성을 제거하기 위한 복수의 데이터 필터링 방법을 제안하고, 소정의 알고리즘(BM25, BM25L 등)을 이용하여 양의 쌍(positive pairs)과 음의 쌍(negative pairs) 데이터를 확보하는 방법을 제안한다.
- [0021] 또한, 본 발명에서는 보다 효과적인 성능을 위해, 코드 표현(code representation)을 사전학습한 언어모델(예컨대, CodeBERT, graphcodebert 등)을 미세조정(finetuning)하고, 유추(inference) 시에 교차 검증 앙상블(Cross-validated ensemble)을 적용할 수 있다.
- [0022] 또한, 단일 모델과 앙상블 모델에 대한 정량적 실험 결과를 도출함으로써 본 발명에서 제안하는 방법의 효과성을 검증한다.
- [0023] 또한, 본 발명에서 제안하는 방법은 적어도 프로세서(processor) 및/또는 메모리(memory)를 포함하는 컴퓨팅 장치에 의해 수행될 수 있다. 컴퓨팅 장치는 PC(Personal Computer), 랩탑 컴퓨터, 태블릿 PC, 서버(server) 등을 포함할 수 있고, 따라서 컴퓨팅 장치는 (프로그래밍 코드의) 유사성 판단 장치로 명명될 수도 있다.
- [0024] 이하에서는, 데이터 전처리 과정, 필터링 전략, 양의 쌍과 음의 쌍을 생성하는 방법, 및 교차 검증 앙상블에 관해 설명한다. 도 1은 본 발명의 전체 개요도를 도시한다.
- [0025] 데이터 전처리(Data Preprocessing)
- [0026] 코드의 특성상 모델의 입력 시퀀스(input sequence)로 입력 하기에 앞서 정제(cleaning) 작업이 수행될 수 있다. 소스 코드가 정리되지 않은 형태로 작성되거나, 소스 코드에 실제 컴파일과는 무관한 텍스트 정보가 포함되어 있을 수 있기 때문이다.
- [0027] 구체적으로, 데이터들(소스 코드들) 각각에 대하여 불필요한 개행(new lines) 및/또는 띄어쓰기가 제거될 수 있다. 일 예로, 불필요한 개행은 (실질적인) 내용이 존재하지 않는 개행을 의미할 수 있으며, 불필요한 띄어쓰기는 각 행의 시작에 위치하는 띄어쓰기, 각 행의 마지막에 위치하는 띄어쓰기, 연속되는 띄어쓰기들 중 어느 하나를 제외한 적어도 하나의 띄어쓰기 등을 의미할 수 있다. 개발자 등의 이해를 돕기 위해 소스 코드에 포함되어 있는 주석(예컨대, '#'으로 표기된 시퀀스) 역시 학습 과정에서는 불필요한 정보이기 때문에 널 스페이스(null space)와 함께 또는 단독으로 제거될 수 있다. 즉, 데이터 전처리 과정은 개행, 띄어쓰기, 주석, 및 널 스페이스 중 적어도 하나를 제거하는 과정을 포함하는 동작으로 이해될 수 있다.
- [0028] 데이터 필터링(Data Filtering)
- [0029] 서로 다른 데이터를 하나의 학습 및 테스트 데이터(training and test data)로 활용하는 과정에서 데이터의 중복성이 발생할 수 있고, 이러한 데이터로 학습한 모델은 정확한 평가를 진행할 수 없다. 추가적인 데이터로 모델의 성능을 향상시키기에 앞서, 추가적인 (학습) 데이터와 테스트 데이터 사이의 데이터 중복성을 관찰하였다.

따라서, 복수의 단계(예컨대, 3 단계)에 걸친 필터링 작업을 수행한다. 3 단계에 걸친 데이터 필터링 방법은 알고리즘 1을 통해 설명한다.

[알고리즘 1]

Algorithm 1 Efficient Data Filtering Strategies

```
A = DATA1, B = DATA2
Load TEST_codes
procedure DEDUPLICATION(sequence, HT)
    → Quick check if i is in HT
    1: Read All A, B values
    2: for i = 0 to len(A[i]) do
    3:     HT.add(A[i])
    → Build HT
    4: for i = 0 to len(A[i]) do
    5:     if B[i] not in HT then
    6:         First_filtered_codes.append(B[i])
    → First Filtering
procedure SIMPLIFY(code)
    7: return “”.join(code.split('\n')).rstrip(' ').strip()
```

```
procedure DELETE INTERSECTION(First_filtered_codes)
    8: for i = 0 to len(First_filtered_codes[i]) do
    9:     if SIMPLIFY(First_filtered_codes[i]) not in
INTERSECTION(TEST_codes) then
    10:
Second_filtered_codes.append(First_filtered_codes[i])
    → Second Filtering
procedure EXHAUSTIVE SEARCH(Second_filtered_codes)
    11: for i = 0 to len (Second_filtered_codes[i]) do
    12:     USE_TOKEN = True
    13:     if Second_filtered_codes[i] in TEST_codes then
    14:         continue
    15:     else
    16:         for s = 0 to TEST_codes do len(s) > 0 and
len(Second_filtered_codes[i]) > and ((Second_filtered_codes[i]
in s)
                or (s in Second_filtered_codes[i]))
    17:         USE_TOKEN = False
    18:         if USE_TOKEN == True then
    19:
Third_filtered_codes.append(Second_filtered_codes[i])
    → Third Filtering
    20: Return Third_filtered_codes
```

1) 데이터의 중복 제거(Deduplication of Data)

먼저, A, B에 해당하는 데이터와 테스트 데이터(TEST_codes)를 호출한다. 이때, A와 B는 동일한 프로그래밍 언어로 작성되면서, 서로 다른 데이터를 나타낸다고 가정할 때, A 및/또는 B의 해시 테이블(Hash Table, HT)을 생성하고, 해시 테이블을 이용하여 A와 B에 포함되어 있는 중복된 데이터를 모두 소거할 수 있다.

일 실시예에 의하면, 첫번째 필터링 기법은 서로 겹치는 시퀀스(sequence) 데이터를 해시 테이블로 필터링하는 방법을 의미한다. HT에 A의 값들을 전부 기록하고, 이러한 값들이 B에 존재하는지의 여부를 확인하는 방법이다. HT는 소스의 단어 단위로 기록된 정보 집합을 의미할 수 있다. 알고리즘 1의 제1 필터링(1 ~ 6 lines)에서, (1) DEDUPLICATION 함수에 입력된 코드 시퀀스와 초기화된 HT는 A로부터 새로운 HT를 생성한다. 또한, (2) 새롭게 생성된 HT를 B와 비교하여 제1 필터링된 코드(First_filtered_codes)를 생성한다(lines 6). 이 과정에서 대부

본의 중복성을 필터링된다. 즉, 제1 필터링된 코드는 중복되는 내용이 삭제된 A와 B를 의미할 수 있다.

2) 교집합의 삭제(Deletion of Intersection)

제2 필터링은, 제1 필터링 과정에서 트레일링 스페이스(trailing space) 등의 이유로 HT로 완전하게 걸러지지 않은 데이터를 걸러주는 것을 목적으로 한다. 걸러지지 않은 데이터는, 문장(sentences)의 왼쪽 끝 또는 오른쪽 끝에 위치하는 공백(white spaces, 예컨대, ' ') 및/또는 탭(tabs, 예컨대, '\t') 등을 의미할 수 있다. 알고리즘 1의 제2 필터링(7 ~ 10 lines)에서, (1) SIMPLIFY 함수로 코드 문자열(character strings)에 존재하는 모든 개행들(new lines)을 전부 이어 붙인다. (2) 문자열의 앞과 뒤에 존재하는 모든 공백과 개행을 제거한다. (3) 이렇게 필터링된 테스트 코드의 시퀀스와 제1 필터링된 코드를 인터섹션(INTERSECTION)함으로써 한 번 더 걸러주는 작업을 수행하여 제2 필터링된 코드(Second_filtered_codes)를 생성한다. 즉, 제2 필터링에서는 문자열에 존재하는 모든 개행을 전부 이어 붙여 앞, 뒤로 존재하는 모든 공백과 문자열 사이의 개행을 소거한 후에 교집합 값만을 따로 걸러주는 작업을 의미한다. 일 예시로, 파이썬(phthon)의 set()을 활용하여, DATA1 = set() 이후, DATA1.intersection(DATA2)를 수행함으로써, 두 DATA1, DATA2로부터 교집합이 되는 데이터만을 반환받을 수 있고, 반환받은 데이터를 제거할 수 있다.

3) 완전 검색(Exhaustive Search)

제2 필터링 과정에서 대부분의 데이터는 제거된다. 그러나, 중복성을 완벽히 제거하기 위한 방법이 필요하다. 알고리즘 1의 제3 필터링에서, 완전 검색을 수행하여 s와 TEST_codes를 상호적으로 비교하면서 몇몇 남아 있는 중복 데이터 흔적들을 제거하고, 최종적으로 학습 데이터로 사용할 제3 필터링된 코드(Third_filtered_codes)를 생성할 수 있다. 즉, 제3 필터링에서는 공백을 기준으로 모든 단어 간의 비교를 통해 중복 데이터를 제거할 수 있다.

양의 쌍과 음의 쌍의 생성(Forming positive & netagive pairs)

코드 유사성 판단 모델을 학습시키기 위해, 모델은 두 가지 종류의 학습 데이터가 필요하다. 하나는 두 코드가 유사하다고 판단할 수 있는 양의 쌍(positive pair)이고, 다른 하나는 두 코드가 유사하지 않다고 판단되는 음의 쌍(negative pair)이다. 양의 쌍은 동일한 문제를 해결할 수 있는(label 0) 코드 쌍을 의미할 수 있고, 두 코드의 토큰 길이의 합은 모델에 입력 가능한 최대 크기인 512를 넘지 않도록 슬라이싱될 수 있다. 음의 쌍은 BM25 알고리즘을 이용하여, 동일한 문제를 해결하지 못하는(label 1) 코드 쌍을 생성할 수 있다.

음의 샘플들의 경우 (신뢰) 스코어((confidence) score)를 기준으로 내림차순으로 정렬 후 순위(ranking)을 매긴다. 이에 본 발명에서는 효율적인 유사도 평가 모델 개발을 위해 문서 검색 방법에서 이미 성능이 검증된 Okapi 시스템의 BM25 알고리즘 및/또는 이를 변형한 BM25L 알고리즘을 활용하여 양의 쌍과 음의 쌍을 생성하는 방법을 제안한다. 이를 기반으로 코드 간 유사성 평가 모델의 효율성을 향상시킬 수 있다.

BM25는 키워드들 $k_1, \dots, k_n$ (n 은 임의의 자연수)을 포함하는 쿼리 문장(query sentence) Q에 대하여 다음 수학식과 같이 문서(document) D의 스코어를 산출할 수 있다.

$$score(D, Q) = \sum_{i=1}^n IDF(k_i) \cdot \frac{f(k_i, D) \cdot (m_1 + 1)}{f(k_i, D) + m_1 \cdot (b \cdot \frac{|D|}{all} + 1 - b)}$$

상기 수학식에서 $f(k_i, D)$ 는 D 내에서 k_i 의 단어 빈도(term frequency, TF)이고, $|D|$ 는 단어들(words)의 길이(length)이고, all 은 문서의 평균 길이(average document length)이다. m_1 과 b 는 하이퍼파라미터들이다.

$$IDF(k_i) = \ln\left(\frac{N - n(k_i) + 0.5}{n(k_i) + 0.5}\right) + 1$$

역 문서 빈도(inverse document frequency, IDF)는 전체 문서 셋(total document set)에서 단어가 드문(rare) 정도를 나타낸다. $IDF(k_i)$ 는 쿼리 텀(query term) k_i 의 IDF 가중치(weight)로 구성되고, $n(k_i)$ 는 k_i 를 포함하는 문서들의 개수를 나타낸다. 상술한 알고리즘을 이용하면, 오리지널 데이터셋의 양이 급격하게 증가할 수 있다.

교차 검증 앙상블 기법(Cross-validated ensemble)

모델의 학습 과정은 각 데이터 간 독립적인 학습 데이터셋(training dataset), 검증 데이터셋(validating dataset), 및 테스트 데이터셋(testing dataset)으로 구분되기 때문에, 데이터의 불균형이 존재한다면 정확한 성능 평가가 이루어질 수 없다. 도 2는 교차 검증 앙상블 기법에 대한 전반적인 개요를 도시한다. 도 2에서, K-

폴드 교차 검증(K-fold cross validation, K는 2 이상의 임의의 자연수로서 예시적인 값은 5임)은 모델 훈련 단계에서 각 데이터셋(datasets)을 K 번씩 K 개의 모델을 학습하므로 더욱 정확한 성능 평가를 수행할 수 있다. 본 발명에서는 각 폴드마다 최고 성능을 가리키는 지점을 기록하여 테스트 단계에서 이렇게 생성된 K 개의 모델의 판단을 앙상블하여 최종 성능을 평가하는 기법을 활용한다.

- [0050] 일 예로, 학습된 모델 결과를 폴드 별로 홀수 개의 결과를 추출할 수 있다. 1개의 모델 당 5-폴드일 경우에 3개의 모델로부터 15개의 결과를 추출할 수 있다. 교차 검증 앙상블 기법을 이용하면, 전체 예측 값의 합을 폴드의 개수로 나눈 결과값을 임계값(threshold)과 비교함으로써 최종 결과를 도출할 수 있다. 구체적으로, 임계값이 0.5일 때, 전체 예측 값의 평균이 0.5를 넘을 경우 1, 그렇지 않을 경우 0으로 예측할 수 있다.
- [0051] 이는 모든 데이터를 학습 과정에 사용함으로써, 모델이 단순히 한 데이터로만 학습하는 것이 아니라 다른 관점에서 학습된 모델의 판단을 통해 더욱 효과적인 성능을 낼 수 있도록 한다.
- [0052] 도 3은 본 발명의 일 실시예에 따른 유사성 판단 모델 생성 방법 또는 유사성 판단 방법을 설명하기 위한 흐름도이다. 도 3에 도시된 각 단계의 적어도 일부는 컴퓨팅 시스템에 의해 수행될 수 있다. 또한, 컴퓨팅 시스템은 유사성 판단 모델 생성 장치나 유사성 판단 장치 등으로 명명될 수도 있다.
- [0053] 우선, 데이터에 대한 전처리 동작이 수행된다(S100). 구체적으로, 복수의 프로그래밍 코드들로 구성된 로 데이터에 대하여, 상술한 전처리 동작이 수행될 수 있다. 다만, 실시예에 따라 S100 단계는 생략될 수도 있다.
- [0054] 다음으로, 데이터에 대한 필터링 동작이 수행될 수 있다(S200). 즉, 프로그래밍 코드 내에서 중복되는 내용에 대한 내용을 필터링함으로써, 중복 데이터를 이용해 학습할 때의 문제점을 예방할 수 있다. 필터링 동작은 3가지 단계로 구성될 수 있고, 이에 대한 상세한 설명은 생략하기로 한다.
- [0055] 필터링된 데이터를 이용하여, 모델의 학습에 이용하기 위한 음의 쌍과 양의 쌍이 생성될 수 있다(S300). 양의 쌍과 음의 쌍은 BM25 및/또는 BM25L 등의 알고리즘을 이용하여 생성될 수 있다.
- [0056] 생성된 음의 쌍과 양의 쌍을 이용하여 소정 모델을 학습 및 미세조정(fine tuning)함으로써, 프로그래밍 코드의 유사성을 판단하는 모델을 생성할 수 있다(S400). 본 발명에서는 사전학습 모델로 graphcodebert와 codebert-mlm를 이용하였으나, 본 발명이 이에 제한되는 것은 아니다. 즉, 임의의 사전학습 모델이 이용될 수 있다.
- [0057] 또한, 교차 검증 앙상블 모델을 이용하여 학습을 진행할 수 있다. 학습 셋과 검증 셋에 변화를 주어 K 번의 학습을 수행할 수 있다. 일 예로, 생성된 학습 데이터를 임의의 기준에 따라 K 개로 분할하고, 분할된 K 개의 데이터를 이용하여 K 개의 모델을 학습하되, 각 모델의 검증 셋이 겹치지 않도록 할 수 있다. 이를 통해 모든 데이터를 학습에 이용할 수 있다.
- [0058] 마지막으로, 입력되는 두 개의 프로그래밍 코드의 유사성이 판단될 수 있다(S500). 유사성은 학습이 완료된 유사성 판단 모델을 이용하여 수행된다.
- [0059] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 컨트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable array), PLU(Programmable Logic Unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(Operation System, OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술 분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(Processing Element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(Parallel Processor)와 같은, 다른 처리 구성(Processing Configuration)도 가능하다.
- [0060] 소프트웨어는 컴퓨터 프로그램(Computer Program), 코드(Code), 명령(Instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(Collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성 요소(Component), 물리적 장치, 가상

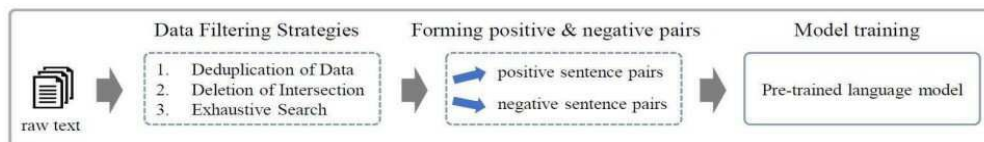
장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 구체화(Embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0061] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

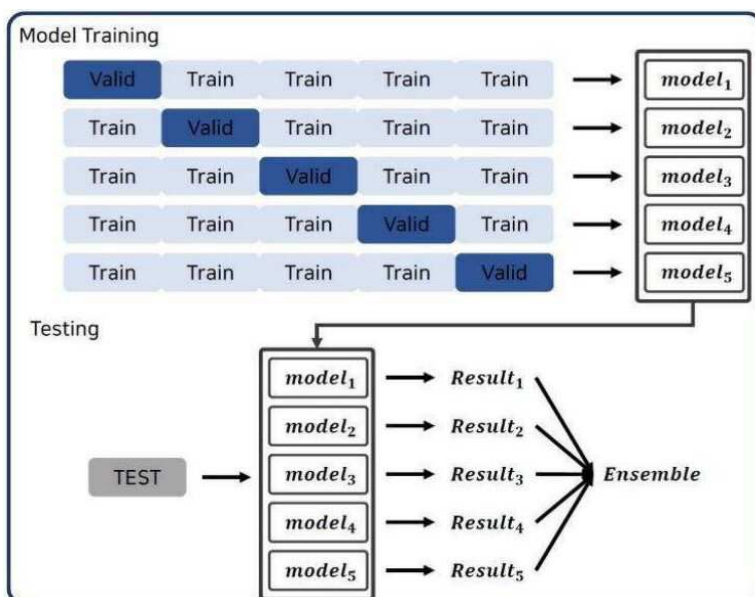
[0062] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성 요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

도면1



도면2



도면3

