JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 19, No. 7, pp. 1357-1363, July. 2018

코드 분포의 선형 회귀를 이용한 프로그램 유사성 분석

임 현 일
경남대학교 컴퓨터공학부

Similarity Analysis of Programs through Linear Regression of Code Distribution

Hyun-il Lim
Department of Computer Engineering, Kyungnam University, Changwon, Gyeongsangnam-do 51767, Korea

[요 약]

정보 기술의 발전과 더불어 인공 지능 및 기계 학습 분야는 다양한 응용 분야에서 성능을 인정받고 있으며, 다양한 응용 분야로 확대되고 있다. 본 논문에서는 기계 학습 방법을 응용한 소프트웨어 분석 방법을 제안한다. 소프트웨어의 특성을 표현하기 위해 소프트웨어의 코드 분포를 분석하고 이 정보를 기계 학습 방법인 선형 회귀를 통해 분석함으로써 유사 소프트웨어를 분석할 수 있는 방법을 제안한다. 소프트웨어의 특성은 프로그램 내에 포함된 명령어에 의해 표현될 수 있으며, 명령어의 분포 정보를 학습 데이터로 활용하였다. 또한, 학습 데이터를 통한 학습 과정은 소프트웨어 유사성 분석을 위한 선형 회귀 모델을 구성한다. 본 논문에서 제안한 방법은 구현 및 실험을 통해 정확성을 검증한다. 본 논문에서 제안한 방법은 소프트웨어의 유사성을 판단할 수 있는 기본 기술로 활용될 수 있을 것으로 기대된다. 또한 기계 학습 방법을 통한 소프트웨어 분석 기술에 응용될 수 있을 것으로 기대된다.

[Abstract]

In addition to advances in information technology, machine learning approach is applied to a variety of applications, and is expanding to a variety of areas. In this paper, we propose a software analysis method that applies linear regression to analyse software similarity from the code distribution of the software. The characteristics of software can be expressed by instructions contained within the program, so the distribution information of instructions is used as learning data. In addition, a learning procedure with the learning data generates a linear regression model for software similarity analysis. The proposed method is evaluated with real world Java applications. The proposed method is expected to be used as a basic technique to determine similarity of software. It is also expected to be applied to various software analysis techniques through machine learning approaches.

색인어 : 선형 회귀, 기계 학습, 유사성 분석, 코드 분석, 코드 분포

Key word : Code analysis, Code distribution, Linear regression, Machine learning, Similarity analysis

http://dx.doi.org/10.9728/dcs.2018.19.7.1357

Received 25 June 2018; **Revised** 18 July 2018
Accepted 25 July 2018

***Corresponding Author; Hyun-il Lim**

Tel: +82-55-249-2650
E-mail: hilim@kyungnam.ac.kr

Ⅰ. 서 론

오늘 날의 현대인은 정보화 사회에서 다양한 디지털 매체들의 홍수 속에서 살아가고 있다. 디지털 정보를 이용하는 일상적인 컴퓨터뿐만 아니라 스마트폰과 같은 통신 기기들도 보편적으로 사용되고 있다. 또한 IT 기술의 발전과 더불어 다양한 새로운 IT 기기들이 개발되고 기존의 시스템과 함께 발전을 거듭하고 있다.

이와 같은 정보화 사회에서 디지털 콘텐츠는 정보의 표현 및 전달 뿐만 아니라 정보를 제공할 수 있는 다양한 기술들과 접목되어 활용되고 있으며, 4차 산업혁명 시대를 맞아 폭발적인 수요 증가를 보여주고 있다. 이와 더불어 기술들을 통합하고 운용할 수 있는 소프트웨어 기술은 점점 더 중요한 역할을 가지고 있다. 정보화 기기의 운용 및 동작을 담당하는 소프트웨어는 IT 기술에서 핵심적인 위치를 차지하고 있으며 소프트웨어 정보를 효율적으로 이해하고 관리하는 기술은 정보화 기술 발전에 중요한 역할을 할 수 있을 것이라 기대된다.

본 논문에서는 정보화 기술에서 소프트웨어가 차지하는 중요성을 인지하고 소프트웨어의 특성을 비교 분석할 수 있는 기술에 대해서 연구한다. 소프트웨어는 바이너리 데이터를 포함하고 있는 디지털 형태로 구성되어 있기 때문에 인간이 이해하고 분석하는 데 어려움이 있다. 또한 소프트웨어가 가지는 구조의 복잡성 때문에 소프트웨어를 이해하고 비교할 수 있는 자동화된 분석 기술에는 어려움이 따른다. 이를 극복하기 위한 다양한 형태의 연구가 진행되고 있는데, 본 연구에서는 인공지능 및 기계 학습의 한 방법으로 적용되는 선형 회귀를 통해 바이너리 소프트웨어를 비교 분석하는 기술을 제안한다.

선형 회귀는 독립 변수로 구성되는 많은 수의 입력 데이터들로부터 특성을 추출하고 상관 관계를 모델링함으로써 결과를 예측할 수 있는 기계 학습의 한 방법이다. 본 논문에서는 바이너리 프로그램의 코드들을 분석을 통해서 선형 회귀를 위한 입력 데이터로 추출하고 이를 선형 회귀를 통한 학습 과정을 거쳐 모델링한 결과로부터 바이너리 프로그램의 비교 결과를 예측할 수 있는 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 사용하는 소프트웨어 유사성 분석에 대한 관련 연구 및 선형 회귀 방법에 대해서 소개하고, 3장에서는 분석하고자 하는 바이너리 프로그램의 코드 분석에 대해서 설명한다. 4장에서는 본 논문에서 제안한 방법의 실험 결과를 보여주고 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

2-1 소프트웨어 유사성 연구

소프트웨어의 중요성이 커짐에 따라 소프트웨어의 특성을 분석하기 위한 다양한 연구가 진행되고 있으며, 특성으로부터 소프트웨어의 유사성을 분석하는 기술은 다양한 응용 분야에서 활용될 수 있다. 일반적으로 소프트웨어의 유사성을 판단하기 위해서 소프트웨어가 가지는 텍스트 특성을 비교할 수 있는데, Yap [1]는 소프트웨어가 가지는 텍스트의 최장 공통 부분수열(longest common subsequence)를 비교하고, 유사성을 분석하는데 활용하였다. k-gram 방법 [2]은 소프트웨어 내의 연속되 k개의 명령어 시퀀스를 비교하고 유사성을 찾아낸다. 소프트웨어가 가지는 내부적인 구조를 비교하는 방법으로 그래프 구조의 특성을 비교할 수 있다. 소프트웨어가 가지는 프로그램 의존성 그래프 (program dependence graph) [3], 전체 프로그램 경로 (whole program path) 그래프 [4], 제어 흐름 그래프 [5] 등에서 소프트웨어가 가지는 특징이 되는 구조를 추출하고 이를 비교하는 방법을 이용하고 있다. 기존의 방법들은 프로그램이 가지는 텍스트 또는 그래프와 같은 특징 정보를 분석을 통해 직접 찾아내고 직접 비교하는 접근 방법을 사용하고 있다. 이런 방법은 프로그램의 복잡도가 커지고 다양한 환경에서 변화하는 프로그램의 특성에 대해 신속하게 분석하고 반영하는데 어려움이 따른다.

본 논문에서는 기계 학습이라는 접근 방법을 이용해서 프로그램의 특성을 데이터로 표현하고 기계 학습 과정에서 프로그램의 특성을 모델링하는 접근 방법을 제안하고 있다. 소프트웨어의 코드 분포 정보는 소프트웨어의 특징을 표현할 수 있으며, 학습을 통한 다양한 환경의 분석 기술에 응용될 수 있을 것이라 기대된다..

2-2 선형 회귀의 개념 및 활용

통계학에서 시작한 선형 회귀는 인공 지능 및 기계 학습 [6,7,8] 기술의 발전과 더불어 데이터의 분포에 대한 학습을 통한 모델링 및 추론을 위한 방법으로 효과적으로 활용되고 있다. 선형 회귀 [6,7,9]는 한 개 이상의 독립 변수에 대해서 독립 변수와의 선형 상관 관계를 데이터 분석 및 모델링을 통해 찾아내는 방법이다. 이 방법은 여러 개의 독립 변수가 있을 때 종속 변수와의 관계를 모델링하고 이를 통해 예측이 필요한 종속 변수의 값을 기존의 데이터로부터 학습을 통한 모델링 결과로부터 추론하는 기법이다.

선형 회귀 방법은 예측하고자 하는 결과가 여러 개의 입력 변수와의 관계로 정량화해서 모델링할 수 있을 때 결과를 예측할 수 있는 분석 방법으로 널리 활용되고 있다. 결과 예측을 위한 모델링에서 일반적으로 최소제곱법을 사용해 회귀 모델을 만들고 예측 결과를 분석한다.

2-3 선형 회귀 이론

선형 회귀 분석 [6,7,9]에 사용되는 데이터는 예측하고자 하는 종속변수 y와 입력으로 사용되는 독립 변수 x에 대해서 주어진 n개의 데이터 집합 $\{y_i, x_{i1}, \cdots, x_{ip}\}_{i=1}^{n}$에 대해 y를

예측할 수 있는 선형 관계를 분석한다. 이 분석 모델은 선형 관계를 가지므로 다음과 같은 수식으로 분석 된다.

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i, \quad i = 1, \cdots, n \quad (1)$$

위 수식에서 y_i는 종속 변수라고 하며 선형 회귀를 통한 예측 모델에서 예측하고자 하는 결과 값에 해당한다. 입력 데이터로 표현되는 x로부터 선형 상관 관계를 분석함으로써 결과를 도출할 수 있는 모델을 디자인할 수 있다.

위 수식에서 $x_{i1}, \cdots, x_{ip}$ 는 독립 변수에 해당한다. 각 변수들 사이에 상관 관계가 없는 독립적인 값을 가지고, 이러한 독립 변수 값들의 쌍을 선형 회귀 분석을 위한 입력 데이터로 사용할 수 있다. 여기서, p는 선형 회귀 분석을 위한 입력 데이터의 개수이다.

위 수식에서 β는 각 입력 데이터 x에 대한 계수 이며, 선형 회귀 분석 모델에서 결과 값을 추론할 수 있는 입력 데이터 x와의 관계를 나타내는 파라미터 계수이다. 그리고, ϵ는 분석을 통해 결과를 추론할 수 있는 오차항으로 입력 데이터와 결과값의 오차를 나타내고, 결과 추론을 위한 오차 보정을 위한 값이 된다.

2-4 최소 제곱법을 이용한 선형회귀 추론

선형 회귀 모델은 입력으로 사용되는 독립 변수와 추론 결과를 가지는 종속 변수간의 상관 관계를 기술하는 모델을 분석하는 방법으로 이를 위한 다양한 기법들이 있다. 본 논문에서는 최소 제곱법을 이용한 추정 방법을 적용해서 선형 회귀 모델을 구성하였다.

최소 제곱법 [9,10]은 추론을 위해 구성되는 모델의 오차를 최소화하기 위해서 오차의 제곱의 합을 최소화하는 접근 방법이다. 각 입력 데이터에 대해서 오차는 실제 결과와의 차이를 의미하는데 추론값과 실제값 사이의 오차가 클수록 추론에 실패할 가능성이 커지게 된다. 따라서 최소제곱법에서는 오차의 제곱을 최소화하는 모델을 분석함으로써 추론 결과의 오차가 실제 결과와 근사한 값으로 추론될 수 있는 모델을 찾고자 한다.

최소 제곱법에서는 일반적으로 결과의 실제값과 분석을 통한 측정값 사이의 차이, 즉 오차의 제곱의 합을 최소화하는 계수를 찾고 이를 이용해서 선형회귀 모델을 구성한다. 실제값을 $\hat{y}$이라고 할 때, 오차 ϵ은 다음과 같다.

$$\epsilon = \hat{y} - y = \hat{y} - (\beta_1 x_1 + \cdots + \beta_p x_p) \quad (2)$$

따라서, 전체 데이터 n 개에 대해서 오차 제곱의 합 SSE는 다음과 같다.

$$SSE = \sum_{i=1}^{n} (\epsilon_i)^2 \quad (3)$$
$$= \sum_{i=1}^{n} (\hat{y_i} - (\beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2$$

따라서, 최소 제곱법을 이용한 선형회귀 모델은 위 수식에서 표현된 오차 제곱의 합 SSE를 최소로 하는 회귀 계수 β를 찾음으로써 모델을 완성할 수 있다.

Ⅲ. 바이너리 코드 분포 분석

3-1 바이너리 코드의 구성

바이너리 파일 [11]은 일반적은 텍스트 파일과 달리 2진수 형태로 데이터를 저장하고 있고, 각 파일 마다 가지는 고유의 형식에 따라서 다양한 정보를 가지고 있다. 이런 바이너리 프로그램의 유사성을 검출하기 위해서는 바이너리 프로그램이 가지는 특징 및 특성값을 효과적으로 추출하고 분석하는 기술이 필요하다. 일반적으로 바이너리 프로그램은 정의된 형식에 따라 컴퓨터에서 실행되는 데 필요한 정보를 포함하고 있다. 이런 정보에는 프로그램의 정보를 알려주는 헤더, 실행에 필요한 명령어, 데이터 값 및 데이터를 저장할 수 있는 공간, 기타 실행에 필요한 정보 들을 포함하고 있다.

그림 1. 자바 클래스 파일의 구조
Fig. 1. The structure of Java classfile

그림 1은 바이너리 코드 중에서 자바 클래스 파일의 구조 [12]를 보여주고 있다. 자바 클래스 파일 [12, 13]은 자바 프로그램의 실행에 필요한 프로그램 헤더 정보, 데이터, 클래스 정보, 메소드 및 명령어 들이 구조에 따라 저장되어 있다.

본 논문에서는 바이너리 프로그램의 분석을 통해서 유사성을 검출할 수 있도록 바이너리 프로그램의 실행 명령어 정보를 분석하고 추출하는 방법을 적용하였다. 바이너리 실행 명령어(opcode)들은 프로그램의 실행 시간에 컴퓨터에서 프로그램을 동작시키기 위한 명령어들을 의미한다. 이러한 명령어들은 프로그램의 주어진 작업을 실행하는 데 필수적인 역할을 하고 있으며, 명령어의 구성 및 실행 순서는 프로그램의 실행 결과를 결정하게 되며 프로그램의 작업 특성을 가장 효과적으로 표현할 수 있는 정보가 될 수 있다.

3-2 바이너리 코드의 분포

바이너리 코드의 실행 정보를 가지는 opcode는 프로그램이 실행할 때 수행되는 가장 기본적인 작업 단위이다. 이 opcode들이 순서대로 실행되면서 프로그램에서 지정한 작업들이 수행된다. 이를 표현할 수 있는 정보들은 opcode의 구성 및 실행 순서, 데이터 등에 의해서 결정될 수 있으며, 본 논문에서는 opcode를 구성하는 명령어들의 분포 정보를 이용한 회귀 분석을 프로그램 유사성 분석을 위한 정보로 활용하였다. opcode 명령어들은 명령어마다 각자 고유의 실행 명령을 가지고 있으며 정해진 작업을 수행하기 위해서 서로 대체하거나 임의로 수정하기 어렵다. 따라서, 프로그램의 유사성 정보를 나타내는 중요한 정보로 프로그램에서 사용되는 opcode의 구성 및 분포 정보를 활용할 수 있을 것이다.

바이너리 opcode의 분포는 프로그램에서 사용될 수 있는 각 명령어별로 프로그램상에 나타나는 횟수를 추출함으로써 구할 수 있다. 이 명령어의 분포는 실제 프로그램의 실행할 때 나타나는 프로그램 실행 시간에서 명령어의 실행 횟수와는 서로 다를 수 있다. 순차적인 구조만을 가지는 프로그램인 경우에는 모든 명령어들이 순서대로 한번씩 수행되기 때문에 프로그램에서 나타나는 명령어의 횟수와 실제 실행 시간에 실행되는 명령어의 회수는 동일하다. 하지만, 대부분의 프로그램에서는 순차 구문이외에 반복 구문, 조건문, 점프 명령 등을 이용해서 제어 흐름을 조절할 수 있기 때문에 실행 시간에 명령어의 실행 회수는 명령어의 분포 정보와 다를 수 있다.

3-3 코드 분포를 이용한 선형 회귀 분석 설계

선형 회귀 분석은 수식 (1)에서 보여준 식과 같이 여러 개의 독립 변수 $(x_1, \cdots, x_p)$ 와 이에 따른 종속 변수 y 를 모델링하고 데이터를 구성함으로써 설계할 수 있다. 본 논문에서는 코드 분포에 대한 선형 회귀 분석을 위해 프로그램에서 사용할 수 있는 각각의 명령어에 대해서 명령어의 분포 정보를 학습 데이터로 추출하고 각 명령어의 추출 횟수를 독립 변수 $(x_1, \cdots, x_p)$ 로 설계하였다. 따라서, 프로그램에서 나타날 수 있는 전체 명령어의 개수가 p라면 각 명령어에 대해서 i번째 명령어의 분포 횟수를 x_i 라고 할 수 있으며, 하나의 프로그램에서 나타나는 전체 프로그램의 명령어 분포는 $(x_1, \cdots, x_p)$ 로 표현할 수 있다.

결과값에 해당하는 종속 변수 y 를 설계하기 위해서 프로그램의 유사성 결과를 추출할 수 있는 정보로 학습 데이터를 설계할 수 있다. 본 논문에서는 서로 유사한 프로그램 사이의 코드 분포 결과에 대해서 $y = 1$ 로 학습하고 서로 다른 프로그램 사이의 코드 분포 결과를 $y = 0$ 로 학습할 수 있도록 설계하였다. 따라서, 2개의 프로그램을 통해서 유사성을 검출하기 위한 선형 회귀 분석의 학습 데이터는 두 프로그램의 코드 분포 비교 결과에 대해서 유사한 프로그램과 유사하지 않은 프로그램으로 분류해서 학습 데이터를 생성하고, 학습을 통해서 선형 회귀 모델을 설계하였다.

Ⅳ. 실 험

4-1 실험 환경의 구성

본 논문에서 제시한 코드 분포에 대한 선형 회귀 및 이를 통한 프로그램의 검출 결과를 실험을 통해 분석하였다. 실험을 위해 구성한 실험 환경은 표 1에서 보여주고 있다.

표 1. 실험 환경
Table 1. Experimental environments

CPU	Core i7 - 2600
RAM	16 GB
operating system	MS Windows 7
Programming language	Python, scikit-learn
Binary code format	Java classfile

본 실험은 MS Windows 7 환경에서 수행되었으며, Python [14]로 테스트 데이터 생성을 위한 바이너리 분석기를 구현하였다. 실험을 위한 입력 데이터는 자바 바이트코드를 포함하는 자바 클래스파일로부터 분석을 수행하였다. 분석기는 자바 클래스파일로부터 자바 바이트 코드 정보를 분석하고 코드의 분포 정보를 생성할 수 있으며, 이를 이용해서 학습 데이터 및 실험 데이터를 생성할 수 있도록 구현하였다. 그리고, Python과 scikit-learn [15]를 이용해서 데이터 학습을 통한 선형 회귀 모델을 생성하는 환경을 구현하였다. 본 논문에서 구현한 시스템은 코드 분포를 통한 유사성을 검출하기 위한 선형 회귀 모델을 생성한다.

4-2 학습을 통한 선형 회귀 모델 생성

본 논문에서 제시한 선형 회귀 분석을 통한 코드 유사성 검출 실험을 위해 코드 분포 데이터를 이용한 학습을 통해서 선형 회귀 모델을 구축하였다.

표 2는 선형 회귀 모델 생성을 위해 사용된 학습 데이터의 정보를 보여주고 있다.

표 2. 선형 회귀 모델 분석을 위한 학습 데이터
Table 2. The learning data for evaluating the linear regression model

Data file	Jakarta ORO 2.0.8
# of classfiles	50
Max # of bytecode	923
Average # of Bytecode	144

선형 회귀 모델 생성을 위한 학습 데이터를 구축하기 위해서 실제 환경에서 사용되는 자바 프로그램 Jakarta ORO [16]을 이용하였다. Jakarta ORO는 자바에서 정규식을 사용할 수 있도록 지원하는 라이브러리로 개발되었다. 본 프로그램은 50개의 클래스파일을 포함하고 있으며, 각 클래스파일은 평균 144개 최대 923개의 바이트코드를 포함하고 있었다. 4-1절에서 구현한 자바 클래스파일 분석기를 통해서 Jakarta ORO 에 대한 코드 분포 분석을 수행하였으며 선형 회귀 모델 생성을 위한 학습 데이터를 생성하였다.

유사성 비교를 위한 학습 데이터 생성을 위해서 자바 바이트코드의 명령어를 변경하고 프로그램의 구조를 변경하는 난독화 도구인 Smokescreen 변환 도구를 사용하였다. Smokescreen [17]은 단순히 프로그램에 사용된 이름만을 변경하는 것이 아니고, 제어 흐름의 변경을 통한 구조를 변경하고 명령어를 바꿈으로써 프로그램의 구조를 변경할 수 있다. 이를 사용함으로써 원래의 자바 프로그램과 동일한 실행 결과를 낼 수 있는 유사 프로그램을 생성할 수 있다. 하지만, 이 프로그램은 명령어의 내용 및 제어 흐름 구조가 변경되기 때문에 단순히 바이너리 코드의 비교를 통해 유사성을 비교하거나 분석하기 어렵다. 변환되어 생성된 자바 클래스파일은 원래의 프로그램과 비교를 통해 유사 프로그램에 대한 코드 분포를 학습할 수 있도록 하였다.

프로그램 내에 포함된 서로 다른 클래스파일 사이에 코드 분포 정보는 서로 다른 프로그램 간의 관계를 표현할 수 있다. 따라서 서로 다른 프로그램과의 비교를 통해서 서로 유사하지 않은 프로그램에 대한 정보를 학습할 수 있도록 하였다.

서로 유사한 프로그램과 유사하지 않은 프로그램 사이의 코드 분포 정보를 학습 데이터로 학습함으로써 선형 회귀 모델을 생성할 수 있었다. 본 절에서 생성한 선형 회귀 모델은 테스트 데이터를 통한 결과를 통해 본 분석의 효율성을 실험할 수 있다.

4-3 유사성 분석 실험 결과

4-2절에서 생성한 코드 분포의 선형 회귀를 이용한 유사성 분석 모델에 대한 효율성을 검증하기 위해서 테스트 데이터에 대한 성능 실험을 수행하였다. 표 3은 본 실험에서 사용한 테스트 데이터의 정보를 보여주고 있다.

표 3. 선형 회귀 모델 생성을 위한 테스트 데이터
Table 3. The test data for generating a linear regression model

Data file	antlr 3.5.2
# of classfiles	117
Max # of bytecode	1,646
Average # of Bytecode	172

테스트 실험에서는 Antlr를 사용하였다. Antlr [18]은 자바로 구현된 구문 분석 생성기 (parser generator)이며, 코드분석을 위해 사용된 자바 클래스 파일은 총 117개로 구성되며 각 클래스 파일에 포함된 바이트코드의 수는 평균 172개 였으며 최대 1,646개의 바이트코드를 포함하고 있었다.

그림 2는 선형 회귀 분석을 통한 코드 유사성 분석 실험 과정을 보여주고 있다. 바이너리 코드 분석기를 통해서 자바 클래스 파일로부터 학습 데이터를 생성하고 이를 이용해서 유사성 분석을 위한 선형 회귀 모델을 생성하고 유사성 분석기를 구성한다. 학습을 통해 구성된 유사성 분석기는 테스트 데이터를 통해서 유사성 분석의 정확성을 검증한다.

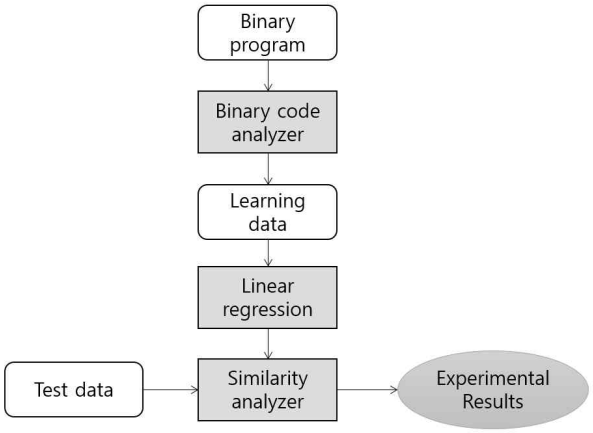

그림 2. 선형 회귀 모델을 통한 유사성 분석 실험 과정
Fig. 2. The experiments for similarity analysis through linear regression model.

표 4는 본 논문에서 제안한 유사성 분석 방법의 실험 결과를 보여주고 있다. 본 실험에서 사용한 전체 테스트 데이터의 수는 13,689개 이며, 그 중에서 유사한 프로그램은 2,711개, 다른 프로그램은 10,978개 이다. 본 실험에서 모델링한 분석 도구로 테스트 데이터에 대한 유사성 예측을 테스트한 결과 총 13,689개의 테스트 데이터 중에서 10,983개의 테스트 데이터에 대해서 정확한 추정이 이루어졌으며, 2,706개의 테스트 데이터에서 오류가 나타났으며, 전체적으로 80.2%의 유사도 예측 정확도를 보여주었다.

표 4. 유사성 분석 실험 결과
Table 4. The experimental results for similarity analysis

# of test data set	13,689
# of similar data	2,711
# of different data	10,978
# of incorrect prediction	2,706
Overall accuracy	80.2 %

본 실험 결과로부터 본 논문에서 제안한 코드 분포의 선형 회귀 모델을 통한 유사성 분석 기법은 높은 수준의 정확도를 가지고 결과를 예측할 수 있는 것으로 평가된다. 연구 결과의 분석 및 지속적인 연구를 통해 분석 정확도의 개선이 이루어질 수 있을 것이다.

선형 회귀를 이용한 학습 모델을 통해 높은 수준의 정확도를 보여주고 있지만 19.8% 정도에서 정확한 추정이 이루어지지 않았다. 정확한 예측이 이루어지지 않은 테스트 데이터에 대해서는 정확한 학습 및 분석 모델이 생성되기 위한 향후 연구를 진행할 계획이다. 향후 연구로는 분석 예측의 정확도를 높이기 위한 선형 회귀 모델의 설계 및 모델 학습 방법에 대한 연구가 필요하다. 또한, 학습 데이터의 표준화 및 정규화를 통해 데이터의 특성을 정확하게 표현할 수 있는 방법에 대해서 연구를 할 예정이다.

프로그램의 유사성을 비교하는 것은 일반적인 텍스트의 비교나 값의 비교처럼 단순한 비교 과정을 통해서 정확한 예측이 어렵다. 따라서, 대부분의 경우에서 자동화된 방법에 한계를 보이고 있으며 사람이 직접 코드를 분석하고 유사성을 찾아야 하는 경우가 많다. 본 논문의 실험 결과에서 기존의 데이터로부터 구성한 선형 회귀 모델은 코드의 분포로부터 높은 정확도로 유사성을 예측할 수 있음을 보여주고 있다. 추가적인 분석 및 지속적인 연구를 통해서 코드 분석의 정확도를 높일 수 있을 것이라 기대되고, 프로그램 비교를 위한 다양한 응용 분야에서 활용될 수 있을 것이다.

Ⅴ. 결 론

정보 통신 기술이 발전함에 따라 다양한 분야에서 소프트웨어를 통한 응용 기술이 개발되고 있다. 또한 소프트웨어가 차지하는 역할은 점점 더 커질 것이라고 기대된다. 소프트웨어의 중요성이 대두됨에 따라 소프트웨어의 의미를 분석하고 특성을 이해할 수 있는 분석 기술은 다양한 응용 분야에서 활용될 것이다. 반면에 소프트웨어가 가지는 본질적인 복잡성 때문에 소프트웨어의 특성을 분석하기 위해 다양한 기술들이 필요하다.

하드웨어의 용량 및 속도가 증가함에 따라 방대한 양의 데이터 연산 및 처리가 가능해졌으며, 이를 바탕으로 인공 지능, 기계 학습 등에 대한 연구가 활발하게 진행되고 있다. 방대한 데이터를 통한 정보 분석 및 활용 기술은 실생활에서 얻을 수 있는 많은 데이터에 대한 회귀 분석 등을 통해 보다 의미 있는 정보를 파악하는데 도움이 될 수 있을 것이다[19]. 본 논문에서는 소프트웨어의 코드 분포를 분석하고 이 정보를 기계 학습의 한 방법인 선형 회귀를 통해 학습 모델을 만듦으로써 프로그램의 유사성을 분석할 수 있는 방법을 제안하고 있다. 소프트웨어의 특성은 프로그램 내에 포함하고 있는 수행되는 명령어에 의해 표현될 수 있으며, 명령어의 분포 정보를 이용해 학습 데이터를 생성하였다. 생성된 학습 데이터를 통해 유사성 비교를 위한 선형 회귀 모델을 생성하고 본 논문에서 제안한 방법의 성능을 검증하였다.

실험을 통해서 본 논문에서 제안한 방법은 소프트웨어의 유사성을 판단할 수 있는 기본 기술로 활용될 수 있을 것으로 기대된다. 향후 연구를 통해서 분석의 정확성을 높일 수 있다면 기계 학습을 통한 소프트웨어 분석에 필요한 기반 기술로 확대될 수 있을 것으로 기대된다.

감사의 글

이 논문은 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2017R1D1A1B03034769).

참고문헌

[1] Michael J. Wise, “Yap3: Improved detection of similarities in computer program and other texts,” In *Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education*, pages 130-134, 1996.

[2] Ginger Myles and Christian Collberg, “k-gram based software birthmarks,” In *Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 314-318, 2005.

[3] Krinke J, “Identifying similar code with program dependence graphs,” In *Working Conference on Reverse Engineering 2001*, pp. 301-309, 2001.

[4] Ginger Myles and Christian Collberg, “Detecting software theft via whole program path birthmarks,” In *International Conference on Information Security* (ISC 2004), LNCS 3225, pp. 404-415, 2004.

[5] Hyun-il Lim, “An Effective Method for Comparing Control Flow Graphs through Edge Extension,” *KIPS Transactions on Computer and Communication Systems*, Vol 2, No. 8, Aug. 2013.

[6] Kevin P. Murphy, Machine Learning: A Probabilistic Perspective, *The MIT Press*, 2012.

[7] Shai Shalev-Shwartz and Shai Ben-David, Understanding Machine Learning: From Theory to Algorithms, *Cambridge University Press*, 2014.

[8] Pedro Domingos, “A few useful things to know about machine learning,” *Communications of the ACM*, Vol. 55, No. 10, pp. 78-87, 2012.

[9] linear regression, Wikipedia [Internet]. Available: https://en.wikipedia.org/wiki/Linear_regression

[10] Least squares, Wikipedia [Internet]. Available: https://en.wikipedia.org/wiki/Least_squares

[11] Binary file, Wikipedia [Internet]. Available: https://en.wikipedia.org/wiki/Binary_file
[12] The class File Format, Java SE Specification, Oracle [Internet]. Available: https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-4.html
[13] Denis N. Antonioli and Markus Pilz, “Analysis of the Java Class File Format,” Technical Report 98.4, Department of Computer Science, University of Zurich, 1998.
[14] Python [Internet]. Available: https://www.python.org/
[15] scikit-learn: Machine Learning in Python [Internet]. Available: http://scikit-learn.org/stable/index.html
[16] The Jakarta-ORO [Internet]. Available: https://jakarta.apache.org/oro/
[17] Smokescreen − Java obfuscator, http://www.javadevelopmentindia.com/technology-amp-integration/technology-amp-integration/obfustication-amp-decompiling/smokescreen/
[18] ANTLR (ANother Tool for Language Recognition) [Internet]. Available: http://www.antlr.org/
[19] Chang-Sik Kim, Su-Jung Choi, Kee-Young Kwahk, “Investigation of Research Trends in Information Systems Domain Using Topic Modeling and Time Series Regression Analysis,” *Journal of Digital Contents Society*, Vol. 18, No. 6, pp. 1143-1150, Oct. 2017.

임현일(Hyun-il Lim)

1995년 : KAIST 전산학과 (공학사)
1997년 : KAIST 전산학과 (공학석사)
2009년 : KAIST 전산학과 (공학박사)

2009년～2010년: KAIST 전산학과 연구원
2010년～현 재: 경남대학교 컴퓨터공학부 부교수
※관심분야 : 소프트웨어 분석, 소프트웨어 보안, 인공 지능, 기계 학습, 소프트웨어 공학, 프로그래밍 언어 등