J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

작성자 분석과 CNN을 적용한 소스 코드 작성자 식별 프레임워크☆

The Identification Framework for source code author using Authorship Analysis and CNN

신 건 윤[1] 김 동 욱[1] 홍 성 삼[1] 한 명 묵[1]*
Gun-Yoon Shin Dong-Wook Kim Sung-sam Hong Myung-Mook Han

요 약

최근 인터넷 기술이 발전함에 따라 다양한 프로그램들이 만들어지고 있고 이에 따라 다양한 코드들이 많은 사람들을 통해 만들어진다. 이러한 측면을 이용하여 특정 작성자가 작성한 코드들 그대로 가져가 자신이 작성한 것처럼 보여주거나, 참고한 코드들에 대한 정확한 표기 없이 그대로 사용하여 이에 대한 보호가 점차 어려워지고 있다. 따라서 본 논문에서는 작성자 분석 이론과 합성곱 신경망 기반 자연어 처리 방법을 적용한 작성자 식별 프레임워크를 제안한다. 작성자 분석 이론을 적용하여 소스 코드에서 작성자 식별에 적합한 특징들을 추출하고 이를 텍스트 마이닝에서 사용하고 있는 특징들과 결합하여 기계학습 기반의 작성자 식별을 수행한다. 그리고 합성곱 신경망 기반 자연어 처리 방법을 소스 코드에 적용하여 코드 작성자 분류를 수행한다. 본 논문에서는 작성자 분석 이론과 합성곱 신경망을 적용한 작성자 식별 프레임워크를 통해 작성자를 식별하기 위해서는 작성자 식별만을 위한 특징들이 필요하다는 것과 합성곱 신경망 기반 자연어 처리 방법이 소스 코드등과 같은 특수한 체계를 갖추고 있는 언어에서도 적용이 가능하다. 실험 결과 작성자 분석 이론 기반 작성자 식별 정확도는 95.1%였으며 CNN을 적용한 결과 반복횟수가 90번 이상일 경우 98% 이상의 정확도를 보여줬다.

☞ 주제어 : 작성자 식별, 작석자 분석, 합성곱 신경망, 기계학습, 코드 분석

ABSTRACT

Recently, Internet technology has developed, various programs are being created and therefore various codes are being made through many authors. On this aspect, some author deceive a program or code written by other particular author as they make it themselves and use other writers' code indiscriminately, or not indicating the exact code which has been used. Due to this makes it more and more difficult to protect the code. In this paper, we propose author identification framework using Authorship Analysis theory and Natural Language Processing(NLP) based on Convolutional Neural Network(CNN). We apply Authorship Analysis theory to extract features for author identification in the source code, and combine them with the features being used text mining to perform author identification using machine learning. In addition, applying CNN based natural language processing method to source code for code author classification. Therefore, we propose a framework for the identification of authors using the Authorship Analysis theory and the CNN. In order to identify the author, we need special features for identifying the authors only, and the NLP method based on the CNN is able to apply language with a special system such as source code and identify the author. identification accuracy based on Authorship Analysis theory is 95.1% and identification accuracy applied to CNN is 98%.

☞ keyword : Author Identification, Authorship Analysis, Convolutional Neural Network, Machine Learning, Code Analysis

1. 서 론

IT산업이 발전함에 따라 다양한 프로그램들이 생겨났으며 해당 프로그램들안에 포함되어있는 다양한 코드들도 만들어지게 되었다. 이러한 코드들은 하루에도 수십개에서 수백 개씩 만들어지고 특정 코드들은 작성자가 무료로 오픈하여 많은 사용자들이 쉽게 접근이 가능하거나, 또는 일정 금액을 지불하고 사용하기도 한다.

이러한 측면을 이용해 특정 작성자가 작성한 코드를

1 Department of Computer Engineering, Gachon University, Seongnam-si, 13120, Korea.
* Corresponding author (mmhan@gachon.ac.rk)
[Received 21 June 2018, Reviewed 25 June 2018(R2 13 August 2018), Accepted 29 August 2018]
☆ 이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2018R1D1A1B07050864)

아무런 표기 없이 그대로 가져와서 마치 자신의 것처럼 사용하거나 일정 금액을 지불하고 사용하는 코드를 불법으로 배포하여 사용하는 경우가 발생하고 있으며 매일 많은 수의 코드들이 만들어지면서 이에 대한 보호가 점차 어려워지고 있다.

이러한 문제를 해결하기 위해 코드 안에 존재하는 작성자를 식별하는 요소들을 찾는 연구가 진행되어 왔으며 기존 연구들은 주로 텍스트 마이닝을 적용하여 작성자를 식별하였으며, 코드의 어휘적, 문자적, 구문론적, 의미론적 특징들을 파악하고 이를 바탕으로 작성자 식별을 수행하였다. 하지만 이러한 연구들에서 사용된 텍스트 마이닝은 해당 문서의 작성자를 분석하는 특징이기 보다는 문서를 분석하는 특징이기 때문에 이를 가지고 소스 코드 안에 숨어있는 작성자 식별 정보를 찾아내는 것은 한계점이 존재한다.

작성자 분석 이론은 텍스트 마이닝 방식과 유사하지만 분석의 주체가 텍스트 마이닝과는 다르게 해당 문서 또는 코드의 작성자이고 분석을 통해 작성자의 스타일을 찾는 것을 목표로 두고 있다. 따라서 본 논문에서는 작성자 분석 이론을 적용한 작성자 식별을 수행하며, 이를 통해 해당 프로그램의 작성자 식별 특징을 찾고 식별하는 연구를 수행한다.

추가적으로 합성곱 신경망을 적용한 작성자 식별을 수행한다. 합성곱 신경망은 데이터 처리시 순차적인 처리를 수행하고 이를 통해 해당 정보들을 분류하기 때문에 글이 들어오는 순서가 중요한 자연어나 코드 등의 데이터를 분류하는데 적합하며 최근 연구들에서는 순환형 신경망과 합성곱 신경망을 적용하여 문장 분류를 수행하거나 텍스트 데이터 셋을 분류를 수행하기도 한다. 하지만 사용되는 데이터들이 소스 코드와 같은 특수한 문장 체계를 가지지 않았으며 짧은 문장들을 대부분을 사용하였다.

본 논문에서는 합성곱 신경망이 가지고 있는 순차적 처리 방식을 적용하여 소스 코드 작성자 식별을 수행하며 이를 통해 합성곱 신경망을 적용한 자연어 처리 기술에 소스 코드와 같은 특수한 문장 체계도 적용이 가능하다는 것과 궁극적으로 코드 작성자 식별이 가능하다는 것을 확인한다.

본 논문에서는 작성자 분석 이론을 통해 코드 안에 존재하는 작성자 식별 요소를 추출하고 이를 텍스트 마이닝을 통해 나온 특징들과 결합하여 작성자 식별을 수행하고 이를 통해 작성자 식별에 사용되는 주요한 특징들을 찾는다. 또한 합성곱 신경망 기반의 자연어 처리 방법을 적용하여 작성자 식별을 수행한다. 이를 통해 작성자 식별을 위한 주요한 특징 연구가 필요하다는 것과 작성자 분석 이론이 이러한 문제점을 해결할 수 있다는 것을 확인하였으며, 합성곱 신경망을 통해서 소스 코드와 같은 특수한 체계를 가지고 있는 문서들도 분석이 가능하다는 것과 이를 통해 코드 작성자 식별이 가능하다는 것을 확인하였다. 또한 실험 결과 작성자 분석 이론 기반 작성자 식별 정확도는 기존의 텍스트 기반 특징들만을 적용하였을 때보다 1.8% 높은 95.1%의 식별 정확도를 보였으며, CNN을 적용한 결과 반복횟수가 90번 이상일 경우 98% 이상의 작성자 식별 정확도를 보여줬다.

2. 관련 연구

2.1 작성자 분석

작성자 분석은 바이너리 및 어셈블리어와 같은 코드들 또는 워드, 한글, 액셀 등과 같은 문서들을 분석하여 해당 문서 안에 존재하는 작성자의 행위, 패턴 및 특징을 찾아 작성자의 특징을 파악하고 식별하는 방법을 의미하며 과거에는 작성자의 스타일을 분류하는 연구를 주로 하였으나 최근 연구에서는 작성된 프로그램의 작성자 스타일 특징을 찾는 연구를 주로 진행하고 있다. 작성자 분석 이론은 작성자 식별, 작성자 특성, 유사 탐지 3가지로 구분할 수 있다.

작성자 식별은 특정 작성자가 작성한 다양한 코드를 분석하여 해당 작성자의 특징을 추출하는 방식을 말하며, 해당 작성자만을 식별 할 수 있는 주요 특징을 추출한다. 작성자 특성은 작성자의 특성을 분석하고 이를 통해 작성자의 프로필을 작성하는 것을 말하며, 유사 탐지는 다양한 코드를 비교 분석하여 한 사람의 작성자가 작성하였는지를 식별하는 방식을 말한다.

작성자 분석에서는 해당 코드의 어휘적(단어, N-gram), 문자적(글자, 숫자), 구문론적(이름 선언 방식, 빈도수), 의미론적(반복 명령어 또는 기능), 응용프로그램적(코드 구조, 사용 언어) 특징등이 사용되며 이를 통해 작성자 식별을 수행한다. 작성자 분석방식에 기계학습을 적용하는 방식은 군집화와 분류 전부 사용이 가능하며 분류를 수행하기 위해서는 수집된 데이터 중에 작성자를 알고 있는 데이터가 있어야한다[1].

접근 방식은 프로필 기반 방식과 인스턴스 기반 방식으로 나뉘며, 프로필 기반 방식은 작성자마다 식별된 프로그램들을 가지고 작성자의 스타일을 추출하여 각 작성자마다 작성자 프로필을 생성하는 방식으로 작성자기 식별되지 않은 코드를 프로필과 비교하여 해당 작성자를

찾는다. 인스턴스 기반 방식은 작성자가 식별된 다양한 프로그램들의 특징을 추출하여 이를 작성자 식별 모델에 적용하는 방식을 의미하며 식별되지 않은 코드에서 나온 특징들과 가장 유사한 방식을 사용한 작성자를 찾는다.

I. Krsul and H. Spafford[2] 연구에서는 작성자의 스타일을 찾아서 분류하는 방식을 제안하였고 코드 안에 들어있는 작성자를 구분하는 특성을 찾아 분류를 수행하였다. G. Andrew[3]에서는 컴퓨터 프로그램에는 알고리즘 레이아웃, 스타일, 환경 등에 따라 다양한 특징이 존재하고 이를 통해 프로그램 작성자가 식별을 수행하였으며, S. Alrabaee et al[4]에서는 작성자를 식별 할 수 있는 주요 특징들을 추출하여 악성코드 분석을 수행하였다.

2.2 소스 코드 특징

소스 코드는 각각의 작성자들이 가지고 있는 특별한 프로그래밍 기법들이 포함되어 있으며, 이러한 다양한 특징들을 가지고 작성자 분석 이론에 적용하여 작성자 식별을 위한 특징들을 추출한다. 소스 코드에서 추출할 수 있는 특징들은 표1과 같다[4,5].

(표 1) 작성자 분석기반 소스 코드 특징들

(Table 1) Source code features based on Authorship Analysis

특 징	정 의
언어적 특징	프로그램 언어에 나타나는 특징
포맷 형식	해당 프로그램 구성 형식
프로그램 특징	특정 컴파일러 및 프로그램만이 가지고 있는 특징 파악
주석 스타일	코드의 포함되어있는 주석 분석
변수명	코드 안에 정의 변수명 분석
단어와 문법	코드에 작성된 단어와 문법을 통해 작성자 분석
버그 및 취약점	반복적인 버그 및 취약점 파악
실행 경로	구조, 주요 키워드 등을 파악
Abstract Syntax Tree	코드 컴파일을 통해 생기는 중간 언어
Control Flow Graph	코드 실행 상태, 경로 등을 표현하기 위해 사용
Program Dependence Graph	프로그램을 나누는 방식

여러 연구에서 소스 코드를 적용하여 작성자 식별을 수행하였으며 S. Alrabaee et al[4]에서는 소스 코드를 통해 일반 작성자와 악성코드 작성자를 식별하는 연구를 하였고 H. Spafford and A. Weeber[5]에서는 소스 코드 분석에 소프트웨어 포렌식 기법을 적용하여 작성자를 식별하는 방법에 대해서 제안하였다.

2.3 합성곱 신경망

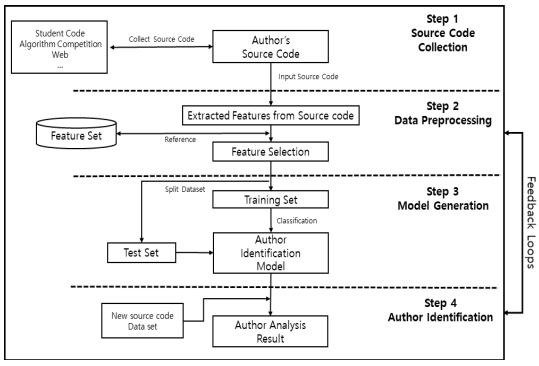
합성곱 신경망은 사람의 신경망을 가지고 구축한 모델로 주로 이미지를 분류하는데 자주사용이 된다. 합성곱 신경망은 합성곱 계층, 폴링 계층과 완전 연결 계층으로 구성되어 있으며 합성곱 계층과 폴링 계층을 반복 수행하여 특징을 추출하고 완전 연결 계층을 통해 분류를 수행한다[6].

합성곱 신경망 기반 자연어 처리는 합성곱 신경망이 가지고 있는 순차적 데이터 처리 방식을 자연어 처리에 적용한 방식으로 수집된 데이터(문서, 코드 등)를 단어 임베딩을 통해 벡터화를 시키고 이를 합성곱 신경망에 적용하여 분류를 수행한다. 합성곱 계층에서 단어 또는 글자의 수를 필터에서 정의한 값에 따라 처리하고 합성곱 연산을 하고 완전 연결 계층을 통해 분류를 수행한다.

M. Moreno and J. Kalita[7]에서는 딥 러닝 방식을 적용한 자연어 처리 프로세스에 대해서 연구하였으며, K.Yoon[8] 연구에서는 합성곱 신경망을 통해서 영화 리뷰 데이터를 분석하였으며, 분석을 통해 긍정적 리뷰와 부정적 리뷰를 분류하였다. Y. Wenpeng et al[9]에서는 합성곱 신경망과 순환형 신경망을 적용한 자연어 처리 프로세스에 대한 연구를 진행하였다.

3. 제안 프레임워크

3.1 작성자 분석 이론기반 작성자 식별



(그림 1) 작성자 분석기반 작성자 식별 프레임워크

(Figure 1) Author identification framework based on Authorship Analysis

(그림 1)은 작성자 분석 이론을 적용한 작성자 식별 프레임워크이다. 해당 방법은 소스 코드 수집, 데이터 전처리, 모델 생성, 작성자 식별 총 4단계로 되어있으며 두 번째 단계인 데이터 전처리 단계에서 작성자 분석 이론을 적용한 특징들을 추출하게 된다.

소스 코드 수집 단계에서는 특정 웹사이트, 알고리즘 구현 대회 등을 통해 코드를 수집하고 데이터 전처리 단계에서는 수집된 코드들을 분석하여 특징 추출 및 선택을 수행한다. 모델 생성 단계에서는 분류 알고리즘을 가지고 모델을 구축하며 작성자 식별 단계에서는 구축된 모델을 가지고 작성자 식별을 수행한다.

특히 데이터 전처리 단계에서는 기존에 작성자 식별에 사용되고 있던 텍스트 기반 특징들과 작성자 분석을 적용한 특징들을 전부 추출하게 되며 해당 특징들을 결합하여 작성자 식별을 수행하고, 추출된 특징들을 특징 셋에 따로 저장을 하게 된다. 해당 특징 셋은 작성자를 식별하는 모델의 사용된 특징들을 변경하게 될 경우, 특징 추출 단계를 다시 거치치 않고 특징 셋 데이터를 가지고 바로 특징 선택을 수행하게 된다.

(표 2) 프로그램 언어별 작성 방식
(Table 2) Writing method by program language

	For	While	If	Switch
C++	for	while	if	switch
Python	for	while	if	dictionary
Java	for	while	if	switch
Scala	for	while	if	pattern matching
Ruby	for	while	if	case expression
	Comment		**Variable**	
C++	//, /* */		char, int, float, double, struct	
Python	#		def	
Java	//, /* */		int, char, short, byte, long, float, double, boolean, class	
Scala	//, /* */		var, val, def, type	
Ruby	#		@, @@, $, def	

본 논문에서는 앞서 서술한 다양한 특징들 중에서 텍스트 기반 특징들과 작성자 분석 이론 기반 특징들을 추출하며 텍스트 기반 특징에는 어휘적, 단어 기반, 구조적 특징으로 구성되어 있고 코드 작성자 스타일은 주석, 변수값, 작성 스타일로 구성되어있다. 두 종류의 특징들을 추출함으로써 작성자를 식별할 수 있는 주요한 특징을 확인하고 이를 통해 작성자 식별 정확도를 높일 수 있다.

텍스트 기반 특징 중, 어휘적 특징에는 코드의 총 문자 수, 총 알파벳 수, 총 공백 수가 있고, 단어 기반 특징은 해당 코드의 총 단어 수, 단어 평균 길이로 구성되어 있으며, 구조적 특징에는 해당 코드의 라인 수가 있다.

작성자 분석 이론 기반 특징들은 텍스트 기반 특징들과는 다르게 다양한 프로그램 언어에서 정의되어있는 명령어 및 변수 선언 방식, 주석 사용 방법, 코드 작성 방법 등이 차이가 존재한다는 것을 인지하고 각각의 프로그램 언어에 맞게 특징을 추출해야 한다. 본 논문에서는 총 5개의 프로그램 언어로 작성된 소스 코드를 사용하였으며, 해당 언어들에서 사용되는 명령어, 주석, 변수 선언 방법에 대해서 연구하였고, S. Alrabaee et al[4]와 H. Spafford and A. Weeber[5]에 나와있는 소스 코드 특징을 참고하여 작성하였다. (표 2)는 본 논문에서 사용하는 프로그램 언어들과 그에 상응하는 작성 방식을 나타내는 표이며 보이는 바와 같이 각 프로그램 별로 조금씩 작성하는 방법이 다르다는 것을 확인 할 수 있다.

(표 3) 본 논문에서 사용하는 특징들
(Table 3) features in proposed framework

유형	종류	이름	정의
텍스트 기반 특징	어휘적 특징	총 글자 수	모든 문자의 수
		총 알파벳 수	모든 알파벳 수
		총 공백 수	모든 공백 수
	단어 기반 특징	총 단어 수	모든 단어의 수
		평균 단어 길이	모든 단어의 평균 길이
	구조적 특징	총 라인 수	코드의 라인 수
코드 작성 스타일	주석 스타일	총 주석 수	모든 주석 수
		주석 작성 길이	주석 작성 유형
	변수값 스타일	변수 이름	변수명 작성 방식
		변수값 작성 스타일	변수명 작성 방식
	작성 스타일	언어	사용된 프로그램 언어
		for_while	for, while 중 선호하는 명령어
		if_switch	if, switch 중 선호하는 명령어

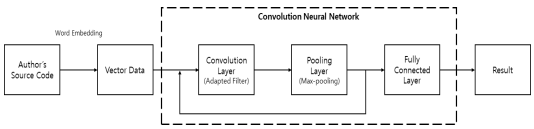
주석 스타일에서 총 주석 수는 코드 작성자가 주석을 자주 사용하는 지를 파악하는데 사용되며, 주석 작성 길이는 코드 작성자가 주석을 작성 할 때 주석의 내용이 단어 위주인지 문장 위주인지를 분석한다.

변수값 스타일에서 변수 이름은 변수 정의 시 변수명을 요약해서 작성하는지 또는 전부 작성하는 지를 확인하고, 변수값 작성 스타일은 변수명 정의 시 소문자만을 사용하는지, 대소문자를 함께 사용하는지, 밑줄 문자를 사용하는 지를 파악하여 작성자의 특징을 추출한다.

작성 스타일에서 언어는 해당 코드가 어느 프로그램 언어를 사용하였는지를 파악하고 for_while과 if_switch은 두 가지 명령어 혹은 해당 프로그램에서 이와 유사한 기능을 하는 명령어 중 어느 것을 더 선호하는 지를 파악한다. 이를 통해 작성자의 작성 스타일을 확인한다. 표 3은 본 논문에서 사용하는 특징들에 대한 설명이 들어있다.

3.2 합성곱 신경망기반 작성자 식별

합성곱 신경망기반의 작성자 식별은 K.Yoon[8]이 제안한 방법을 참고하여 연구를 진행하였으며, (그림 2)과 같은 순서로 수행한다. 수집된 소스 코드를 단어 임베딩을 통해서 벡터화된 데이터로 변환하고 이를 합성곱 신경망에 적용하여 분류를 수행한다.



(그림 2) 합성곱 신경망을 적용한 작성자 식별 프레임워크
(Figure 2) Author identification framework based on Convolutional Neural Network

합성곱 계층에서는 벡터화를 진행한 소스 코드 데이터에 필터를 적용하여 정의한 N의 값에 따라 합성곱 연산을 수행하고 폴링 계층에서는 최대값 폴링을 적용하여 차원 축소를 수행한다. 최대값 폴링은 합성곱 연산을 통해 나온 값들 중 정의한 범위 내에서 가장 높은 값을 추출하는 방식이다. 완전 연결 계층에서는 반복적인 합성곱 계층과 폴링 계층을 통해 추출된 특징을 가지고 분류를 수행한다. 해당 계층을 통해서 소스 코드 중 특정 작성자가 작성한 소스 코드와 그 외의 다른 작성자들이 작성한 소스 코드를 분류한다.

4. 실 험

4.1 실험 환경

실험에 사용된 하드웨어 및 OS 환경은 다음과 같다.

- CPU : Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
- RAM : 16.0 GB
- OS : Window 10 64bit

실험에 사용된 tool은 Python[10]을 사용하였고 버전은 3.6.5이다. 분류 알고리즘을 적용하기 위한 패키지는 scikit-learn[11]을 사용하였으며 버전은 0.19.1이다.

4.2 실험 데이터

본 논문에서는 Google Code Jam[12]과 Github[13]에 있는 소스 코드를 사용하여 작성자 분석을 수행하며 실험에 사용하는 데이터에 대한 설명은 표 4에 나와있다. 총 5명의 작성자가 작성한 271개의 소스 코드이며 사용한 프로그램 언어는 5개로 구성되어있다.

(표 4) 실험 데이터 셋
(Table 4) Experiment data set

작성자명(가명)	보유 소스 코드 개수(개)
a	51
b	76
c	42
d	40
e	62

4.3 작성자 분석기반 작성자 식별 실험

본 연구에서는 작성자 분석을 적용한 특징들과 기존에 사용하였던 특징들을 결합하여 작성자 식별을 수행한다. 작성자 식별 정확도를 측정하기 위해서 TP(True Positive), TN(True Negative), FP(False Positive), FN(False Negative)을 통해 precision과 recall값을 구하고 이를 가지고 조화평균을 구한다. SVM, DT, RF, KNN을 적용하여 작성자 식별하고 k-fold 교차 검증을 통해 분류 알고리즘을 평가하였으며 평가 시 k의 값은 10으로 설정하였다.

실험은 아래 작성된 바와 같이 2가지를 진행하며, 이때 N-gram의 N값은 3으로 설정하여 실험을 수행하였는데, 이는 N의 값을 다양하게 적용하여 실험한 결과 N의 값이 3일 때 가장 우수한 성능을 보였기 때문이다. 텍스트 기반 특징들은 텍스트 마이닝 기법을 적용하여 추출한 특징들을 의미한다.

- N-gram과 텍스트 기반 특징을 적용한 작성자 식별
- N-gram 및 텍스트 기반 특징과 작성자 분석 이론 기반 특징을 적용한 작성자 식별

4.3.1 N-gram과 텍스트 기반 특징을 적용한 작성자 식별

첫 번째 실험인 N-gram과 텍스트 마이닝을 적용한 특징 기반 작성자 식별 실험에서는 (표 5)와 같은 실험 결과가 나왔으며 SVM을 적용하였을 때 92.7%의 작성자 식별 정확도를 보였다.

(표 5) N-gram과 텍스트 기반 특징을 적용한 실험 결과
(Table 5) Results of an experiment with N-gram and text-based feature

Algorithm	accuracy	precision	recall	F1-measure
SVM	**0.9273**	**0.9418**	**0.9272**	**0.9240**
KNN	0.8545	0.8545	0.8545	0.8545
DT	0.9091	0.9137	0.9090	0.9079
RF	0.8364	0.8429	0.8363	0.8368

4.3.2 N-gram 및 텍스트 기반 특징과 작성자 분석 이론 기반 특징을 적용한 작성자 식별

앞선 실험에서 사용된 특징들과 작성자 분석 이론을 적용한 특징들을 결합한 작성자 식별 실험의 실험 결과는 (표 6)과 같으며 SVM이 95.1% DT가 94.5%의 작성자 식별 정확도를 보였다.

두 가지 실험을 통해 코드 작성자 스타일을 결합한 데이터 셋의 경우 전보다 더 높은 식별 정확도를 보인 것을 확인하였으며 텍스트 마이닝기반 특징과 코드 작성자 스타일 실험 비교결과 코드 작성자 스타일을 결합하였을 때 1.8% 더 높은 식별 정확도를 보인 것을 확인하였다. 이를 통해 제안하는 프레임워크가 작성자 식별에 적용 가능하다는 점과 기존 방식보다 작성자 식별에 효과적인 것을 확인하였다.

(표 6) N-gram 및 텍스트 기반 특징과 작성자 분석 이론 기반 특징을 적용한 작성자 식별
(Table 6) Results of an experiment with N-gram, text-based feature and Authorship Analysis based feature

Algorithm	accuracy	precision	recall	F1-measure
SVM	**0.9512**	**0.9587**	**0.9512**	**0.9491**
KNN	0.7857	0.8071	0.7857	0.7727
DT	**0.9455**	**0.9477**	**0.9454**	**0.9454**
RF	0.8364	0.8337	0.8363	0.8343

4.4 합성곱 신경망기반 작성자 식별 실험

합성곱 신경망을 적용하여 작성자 식별을 수행하였으며 수집한 데이터 셋을 특정 작성자와 그 외 작성자 그룹으로 분류하여 실험을 수행하였고 합성곱 계층과 폴링 계층이 10번씩 반복될 때마다 식별 정확도를 측정하였다.

실험결과는 (표 7)과 같이 나타났으며, 반복횟수가 90 이상일 때부터 식별 정확도가 최대치가 되었으며 이를 통해 자연어 처리를 위한 합성곱 신경망을 코드 작성자 식별에 적용이 가능하다는 것을 확인 하였다.

(표 7) 합성곱 신경망을 적용한 실험 결과
(Table 7) Convolutional Neural Network applied experiment result

반복횟수	a	b	c	d	e
10	0.545	0.615	0.580	0.430	0.605
20	0.910	0.850	0.905	0.955	0.895
30	0.910	0.965	0.905	0.955	0.915
40	0.965	0.995	0.905	0.955	0.945
50	0.980	0.985	0.915	0.965	0.975
60	0.995	1.000	0.935	0.970	0.995
70	1.000	1.000	0.980	0.980	0.995
80	1.000	1.000	0.985	0.985	1.000
90	**1.000**	**1.000**	**1.000**	**0.985**	**1.000**
100	1.000	1.000	1.000	0.990	1.000

4.5 특징 분석 실험

추가적으로 작성자 분석 이론 기반 작성자 식별 실험과 합성곱 신경망 기반 작성자 식별 실험에 사용된 특징들을 분석하여 작성자 마다 얼마나 다른 특징 값을 가지고 있는 지를 분석하는 실험을 수행하였다.

분석 방법은 해당 특징들의 평균값을 구하여 이를 비교 분석하는 방식으로 진행하였으며, 평균값을 구할 수 없는 사용 프로그램 언어와 같은 특징들은 주로 사용되는 언어 또는 선언방식으로 작성하였다. 또한 for_while과 if_switch 특징은 해당 값이 1에 가까울수록 for과 if의 비율이 높다는 것을 의미한다.

결과값은 (표 8)과 같으며 텍스트 마이닝 기반 특징에서는 5명의 작성자가 가지고 있는 특징들이 유사한 값을 가지고 있는 것으로 확인이 되었으나, 작성자 분석 기반 특징들에서는 작성자마다 각각의 특성을 가지고 있다는 것이 확인되었다. 특정 작성자들은 한 가지 프로그램 언어만 쓰기 거나 혹은 두 가지 이상의 언어를 섞어서 사용

하기도 하였으며, 주석을 달지 않는 작성자가 있는 반면 모든 코드에 주석을 작성하여 해당 코드들에 대한 설명을 명시해 놓은 작성자도 있었다. 또한 변수 선언 방식에서도 특정 작성자는 변수명 선언시, 해당 문자 전부를 사용하거나 요약형을 사용하는 등의 차이를 보였으며, 작성자 소문자, 대문자 혹은 밑줄 문자를 선호하는 것에 대해서도 많은 차이를 보였다.

(표 8) 특징 분석 결과
(Table 8) Feature analysis result

텍스트 기반 특징						
	총 글자 수	총 알파벳 수	총 공백 수	총 단어 수	평균 단어 길이	총 라인 수
a	166.72	4.29	1329.31	113.31	24.68	77.82
b	170.90	4.92	1482.92	110.00	26.66	60.85
c	186.85	5.47	1551.87	153.68	15.14	70.21
d	119.12	4.55	1105.15	84.32	14.90	57.00
e	72.00	4.44	654.01	48.24	9.16	29.40

작성자 분석 이론 기반 특징							
	총 주석 수	주석 작성 길이	변수 이름	변수값 작성 스타일	언어	for_while	if_switch
a	6.90	35.86	전체	대문자	Python	0.86	0.23
b	15.45	42.09	전체	대문자, 밑줄 문자	Python	0.89	0.14
c	1.48	34.49	전체	밑줄 문자	Java, Scala	0.88	0.61
d	0.80	2.26	요약	대문자	C++	0.89	0.68
e	0.00	0.00	요약	대문자	C++	0.91	0.46

이를 통해 작성자 분석 이론을 적용한 특징이 작성자를 분류하고 분석하는데 기존의 방식인 텍스트 기반 특징보다 효과적이라는 것을 확인할 수 있었고 또한 분석하는 작성자의 수와 관계없이 해당 작성자가 가지고 있는 고유한 특성(주석 작성 성향, 사용 언어, 변수명 선언 방식 등)을 찾을 수 있다는 것을 확인할 수 있었다. 이를 통해 각각의 작성자 마다 주로 선호하는 작성 방식을 쉽게 파악할 수 있었다.

5. 결 론

작성자 분석 이론을 적용한 작성자 식별 연구와 특징 분석 연구를 통해서 작성자 분석 이론을 적용한 특징이 기존에 주로 사용하였던 텍스트 기반 특징보다 작성자를 식별하는데 더 좋은 성능을 보인다는 것을 확인하였으며, 특징 분석을 통해 작성자 분석 이론기반 특징들이 작성자를 식별 성능향상에 도움이 되는 주요한 특징들이라는 것을 확인할 수 있었다. 또한 합성곱 신경망을 적용한 작성자 식별 연구를 통해서도 현재 합성곱 신경망의 특징인 순차적 처리를 적용한 자연어 처리 방법이 소스 코드와 같은 특수한 체계를 가지고 있는 언어에서도 사용될 수 있다는 것과 이를 통해서 작성자 식별이 가능하다는 것을 확인하였다.

또한 실험 결과 작성자 분석 이론 기반 식별 정확도는 텍스트 기반 특징들만을 적용하였을 때보다 1.8% 높은 95.1%의 식별 정확도를 보였으며, CNN을 적용한 결과 반복횟수가 90번 이상일 경우 98% 이상의 작성자 식별 정확도를 보여줬으며, 실험을 통해 작성자 분석 이론과 합성곱 신경망이 작성자를 식별을 수행 할 수 있고 작성자 분석 이론 특징들이 작성자를 판별하는 주요한 특징이 된다는 것을 확인하였다. 이는 작성자 분석 이론과 합성곱 신경망을 적용한 작성자 식별 연구를 통해 기존과 또 다른 방향을 제안함으로써 기존에 작성자 식별 연구가 가지고 있던 관점을 넓혀주고 작성자 식별을 위한 각각의 작성자 프로필(Author Profile) 작성 시, 해당 특징들이 작성자를 정의하는데 보다 정확한 정보를 제공할 수 있다는 것과 합성곱 신경망을 통해 작성자 식별을 수행함으로써 최근 많은 연구가 진행되고 있는 인공 신경망을 작성자 식별 연구에 적용할 수 있다는 것을 확인하였다.

하지만 실험에 사용된 작성자의 수가 적어 해당 작성자들을 구분하는 특징들 사이의 격차가 크게 나타나고, 특징들의 수가 적어 작성자 식별을 수행하는데 한계점이 존재한다. 향후 이를 극복하기위하여 더 많은 작성자와 소스 코드를 수집하여 작성자 식별연구를 진행함으로써 이러한 한계점을 극복하고 작성자 분석 이론을 기반으로 한 다양한 특징들을 연구하여 제안한 작성자 식별 프레임워크에 적용할 것이다. 본 논문에서는 소스 코드만을 사용하여 작성자 식별을 수행하였지만 바이너리 코드 어셈블리 코드 등과 같은 다양한 코드를 통한 작성자 식별 연구를 진행하여 다양한 측면에서 작성자 식별 연구를 진행할 것이다. 또한 합성곱 신경망에 작성자 분석기반 특징을 적용하여 작성자를 식별하는 연구를 진행하고 이를 통해 궁극적으로 악성코드의 소스 코드 및 바이너리 코드 등을 분석하여 공격자 및 공격그룹을 식별하는 연구를 진행할 것이다.

참고문헌(Reference)

[1] E. Stamatatos, “A Survey of Modern Authorship Attribution Methods”, American Society for Information Science and Technology, Vol 60, Issue 3, pp 538-556, 2009.
https://doi.org/10.1002/asi.21001

[2] I. Krsul, H. Spafford, ″Authorship Analysis: identifying the author of a program″, Computer & Security, pp 233-257, 1997.
https://doi.org/10.1016/0167-4048(96)81683-x

[3] G. Andrew, S. Philip, M. Stephen, “Software Forensics Extending Authorship Analysis Techniques to Computer Programs”, Information Science, 1997.
http://hdl.handle.net/10523/872

[4] S. Alraba, P. Shirani, M. Debbabi, L. Wang, ″On the Feasibility of Malware Authorship Attribution″, Foundations and Practice of Security, pp 256-272, 2016.
https://doi.org/10.1007/978-3-319-51966-1_17

[5] H. Spafford, A. Weeber, ″Software Forensics Can We Track Code to its Authors?″, Computers & Security, Vol 12, issue 6, pp 585-595, 1993.
https://doi.org/10.1016/0167-4048(93)90055-a

[6] D. Britz, “Understanding Convolutional Neural Networks for NLP”, WILDML, 2015.
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[7] M. Moreno, J. Kalita, ″Deep Learning applied to NLP″, arXiv, 2017.
https://arxiv.org/abs/1703.03091

[8] Y. Kim. ″Convolutional Neural Networks for Sentence Classification”, Empirical Methods on Natural Language Processing, 2014.
https://doi.org/10.3115/v1/d14-1181

[9] W. Yin, K. Kann, M. Yu and H. Schütze, ″Comparative Study of CNN and RNN for Natural Language Processing″, arXiv, 2017.
https://arxiv.org/abs/1702.01923

[10] Python, ″https://www.python.org/″

[11] scikit-learn, ″http://scikit-learn.org/stable/″

[12] Google Code Jam, “https://code.google.com/codejam/”

[13] Github, “https://github.com/”

[14] S. Burrows, M. Tahaghoghi, “Source Code Authorship Attribution using n-grams”, In Proc. of the Australasian Document Computing Symposium, 2007.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.68.5920

[15] J. Houbardas and E. Stamatatos, “N-gram Features Selection for Authorship Identification”, AIMSA, pp 77-86, 2006.
https://doi.org/10.1007/11861461_10

[16] J. Kothari, M. Shevertalov, E. Stehle, S. Mancoridis, “A Probabilistic Approach to Source Code Authorship Identification”, Information Technology, 2007.
https://doi.org/10.1109/itng.2007.17

[17] A. Caliskan, F. Yamaguchi, E. Dauber, R. Harangm K. Rieck, R. Greenstadt and A. Narayanan, ″When Coding Style Survives Compilation: De-anonymizing Programmers from Executable Binaries”, arXiv, 2016.
https://doi.org/10.14722/ndss.2018.23304

[18] G. Frantzeskou, G. MacDonell and E. Stamatatos, “Source code authorship analysis for supporting the cybercrime investigation process”, INSTICC, pp 85-92, 2004.
https://doi.org/10.5220/0001390300850092

[19] N. Rosenblum, P. Miller and X. Zhu, ″Recovering the Toolchain Provenance of Binary Code″, International Symposium on Software Testing and Analysis, pp 100-110, 2011.
https://doi.org/10.1145/2001420.2001433

[20] N. Rosenblum, X. Zhu and B. Miller, ″Learning to Analyze Binary Computer Code″, AAAI Conference on Artificial Intelligence, 2008. Computer Security – ESORICS, pp 172-189, 2011.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.146.1395

[21] N. Rosenblum, X. Zhu and B. Miller, ″Who wrote this code? identifying the authors of program binaries″, Computer Security - ESORICS, 99 172-189, 2011.
https://doi.org/10.1007/978-3-642-23822-2_10

[22] M. Barreno, B. Nelson, D. Joseph and D. Tygar, ″The security of machine learning″, Machine Learning, Vol 81, Issue 2, pp 121-148, 2010.
https://link.springer.com/article/10.1007/s10994-010-5188-5

[23] D. Joseph, L. Pavel, R. Fabio, J. Doug, N. Blaine, "Machine Learning Methods for Computer Security", Dagstuhl Perspectives Workshop, 2013.
[24] A. Abbasi and H. Chen, "Applying authorship analysis to extremist-group web forum messages", IEEE Intelligent Systems, Vol 20, Issue 5, pp 67-75, 2005. https://doi.org/10.1109/mis.2005.81

◐ 저 자 소 개 ◑

신 건 윤(Gun-Yoon Shin)
2017년 가천대학교 인터랙티브 미디어 융합학과(공학사)
2018년 가천대학교 일반대학원 컴퓨터공학과(공학석사)
2018년~현재 가천대학교 컴퓨터공학과 박사과정
관심분야 : 디지털 포렌식, 악성코드 분석, 공격자 식별, 기계 학습
E-mail : bobo7754@naver.com

김 동 욱 (Dong-Wook Kim)
2015년 가천대학교 컴퓨터공학과(공학사)
2017년 가천대학교 일반대학원 컴퓨터공학과(공학석사)
2017년~현재 가천대학교 컴퓨터공학과 박사과정
관심분야 : Data Mining, 인공지능, Data fusion, Anomaly Detection
E-mail : kog7306@naver.com

홍 성 삼(Sung-Sam Hong)
2009년 가천대학교 전자거래학과(공학사)
2011년 가천대학교 일반대학원 전자계산학과(공학석사)
2016년 가천대학교 일반대학원 전자계산학과(공학박사)
2016년~현재 가천대학교 컴퓨터공학학과 연구교수
관심분야 : 정보보호, 인공지능, 데이터 마이닝, 데이터 분석, 지능형 시스템
E-mail : sungsamhong0@gachon.ac.kr

한 명 묵(Myung-Mook Han)
1980년 연세대학교 공과대학(공학사)
1987년 뉴욕공과대학교 대학원 컴퓨터공학과(공학석사)
1997년 오사카시립대학교 대학원 정보공학부(이학박사)
1998년~현재 가천대학교 컴퓨터공학과 교수
관심분야 : 정보보호, 알고리즘, 데이터 마이닝
E-mail : mmhan@gachon.ac.kr