KIPS Trans. Softw. and Data Eng.
Vol.8, No.9 pp.373~378 pISSN: 2287-5905



https://doi.org/10.3745/KTSDE.2019.8.9.373

# Souce Code Identification Using Deep Neural Network

Jisu Rhim† · Tamer Abuhmed††

## ABSTRACT

Since many programming sources are open online, problems with reckless plagiarism and copyrights are occurring. Among them, source codes produced by repeated authors may have unique fingerprints due to their programming characteristics. This paper identifies each author by learning from a Google Code Jam program source using deep neural network. In this case, the original creator's source is to be vectored using a pre-processing instrument such as predictive-based vector or frequency-based approach, TF-IDF, etc. and to identify the original program source by learning by using a deep neural network. In addition a language-independent learning system was constructed using a pre-processing machine and compared with other existing learning methods. Among them, models using TF-IDF and in-depth neural networks were found to perform better than those using other pre-processing or other learning methods.

Keywords : Computer Forensic, Frequency Based Embedding, TF-IDF, Deep Learning, CNN

# 심층신경망을 이용한 소스 코드 원작자 식별

임 지 수† · Tamer Abuhmed††

## 요 약

현재 프로그래밍 소스들이 온라인에서 공개되어 있기 때문에 무분별한 표절이나 저작권에 대한 문제가 일어나고 있다. 그 중 반복된 저자가 작성한 소스코드는 프로그래밍 특성상 고유의 지문이 있을 수 있다. 본 논문은 구글 코드 잼 프로그램 소스를 심층신경망을 이용한 학습을 통해 각각의 저자를 분별하는 것이다. 이 때 원작자의 소스를 예측 기반 벡터나, 주파수 기반 접근법인 TF-IDF등의 전처리기를 사용하여 입력값들을 벡터화해주고, 심층신경망을 이용한 학습을 통해 각 프로그램 소스 원작자를 식별하고자 한다. 전처리기를 이용하여 언어에 독립적인 학습시스템을 구성하고, 기존의 다른 학습 방법들과 비교하였다. 그 중 TF-IDF와 심층신경망을 사용한 모델은 다른 전처리기나 다른 학습방식을 사용한 것보다 좋은 성능을 보임을 확인하였다.

키워드 : 컴퓨터 법의학, 예측 기반 벡터, TF-IDF, 심층 학습, CNN

## 1. 서 론

현재 많은 프로그램들이 무료로 공개되고 있고, 여러 소프트웨어들은 다양한 오픈소스를 사용하고 있다. 그 중에서는 무분별하게 본인이 작성하지 않은 소스코드를 가져다 쓰고 있다. 이로 인해 다양한 분야에서 소프트웨어 표절이 점점 더 보편화되고 있기 때문에 소프트웨어 저자 판별의 중요성이 조금씩 커지고 있다. 그 외에도 소프트웨어에서 저자 판별은 악성코드 개발자를 추적하거나 Free and Open-Source Software(FOSS) 같은 라이선스가 있는 오픈소스의 표절 판별에도 도움이 될 수 있다. 소스 코드는 기본적으로 각 단어를 키워드로 사용할 수 있는 텍스트로 간주 할 수 있는데, 소스코드의 표현은 기능적으로 제한되어 상용어로 이루어진 텍스트보다 한계가 있다. 그럼에도 불구하고 프로그래머는 여전히 소스 코드에 지문을 남긴다. 예를 들어, 프로그래머는 특정 키워드들을 습관적인 코드 조각으로써 여러번 사용하게 된다. 이러한 소스코드의 지문[1, 2]들을 반복하여 학습할 수 있다면, 소프트웨어의 출처를 추적하고 프로그램 작성자를 식별할 수 있다. 소스코드에 대한 지문의 검사는 기존에는 소스코드의 어휘적, 구조적, 구문적인 특징을 직접 선별하여 표현했으나 선택적 가중치를 통한 예측기반벡터나 주파수 기반 접근법을 통한 전처리기를 통한 맵핑으로 소스코드의 지문을 학습하여 구분하려고 한다. 또한 기존의 소스코드 식별법은 주로 프로그래머들의 작성 방법으로 이루어지는데, 이때 프로그래밍언어에 따라 결과가 달라진다. 각각의 프로그래밍 언어의 키워드들이 동일하지 않은 언어의 소스코드 식별법을 활용하

※ 이 논문은 과학기술정보통신부의 재원으로 한국연구재단의 지원을 받아 연구되었음(NRF-2016R1D1A1A03934816).
† 정 회 원 : 인하대학교 컴퓨터공학과 석사
†† 정 회 원 : 인하대학교 컴퓨터공학과 교수
Manuscript Received : January 1, 2019
First Revision : May 29, 2019
Accepted : July 30, 2019
* Corresponding Author : Tamer Abuhmed(tamer@inha.ac.kr)

기 위해서는 언어의 특성을 고려해야 하는 불편함이 있다. 이러한 문제를 해결하기 위해서 본 논문에서는 전처리기를 활용하여 텍스트를 벡터화한 후 심층신경망을 이용한 프로그램밍 언어 독립적인 프로그래머 식별 기법을 제안한다.

본문에서는 기존의 연구와 차별화하여 전처리기를 사용한 후 심층신경망을 이용한 학습 처리 모델을 설계하고 구현한다. 기존에 쓰이던 방식은 소스코드에서 N-gram씩 직접 어휘, 구조, Abstract Syntax Tree(AST)로 나온 특징을 추출하거나 트리형식으로 바꾸었다. 그러나 이런 방법은 소스코드의 특성이 프로그래밍 언어마다 일치하지 않고, 직접 특징을 추출해줘야 되기 때문에 우리는 소스코드의 키워드들을 선별하여 전체 코퍼스 대비 벡터화하거나 예측 기반 벡터 모델을(Frequency based Embedding) 사용하여 다른 언어라도 활용 가능하도록 전처리기를 만들었다. 전처리기 방법 중에서는 각각의 소스코드의 키워드들을 벡터화하여 정리하는 임베딩(Embedding)을 사용한다. 이 때 문서들과 단어들의 특징을 비교하기 위해, 키워드들의 빈도수를 전체 소스코드 길이와 비교하여 나타내는 Term Frequency - Inverse Document Frequency(TF-IDF)[3]를 매핑하여 매트릭스로 만들거나 Skip-gram[4] 모델같은 예측 기반 벡터를 사용한다. 전처리기를 통과하여 나타낸 매트릭스는 심층신경망[5]을 사용하여 각각의 소스코드에 대한 특징을 학습한 후에 원저자를 판별한다. AST를 이용하여 만든 결정트리나[2] 트리를 기반으로 하여 특징을 뽑아낸 실험[6]과 다르게 모든 언어를 같은 시스템에 사용할 수 있기 때문에 언어 독립적으로 3가지 언어를 사용하여 실험을 진행한다. 관련 방법에 대해서는 2장에서 설명하고 전체적인 구조에 대해서는 3장에서 설명한다. 실험 결과와 실험에 대한 파라미터(parameter)들은 4장에 다룬다. 5장은 실험에 대한 내용을 바탕으로 결론을 이야기한다.

## 2. 관련 연구 및 배경

소스코드 저자 식별에 대한 연구는 일반적으로 어휘, 문법, 구조적인 특징을 찾아 분류하는 방법으로 진행되고 있다. 대표적인 소스코드 저자 식별 방법은 N-gram을 통한 분류 기법이 있다. N-gram에서 n은 연속적 시퀀스이며 바이트, 문자 또는 단어단위의 개수인데, 바이트, 문자 및 워드에서 N-gram은 텍스트 저작자 표시, 음성 인식, 언어 모델링, 상황에 맞는 맞춤법 교정, 광학 문자 인식 등과 같은 다양한 응용 프로그램에서 사용된다[7]. Georgia Frantzeskou 등[8]의 연구에서는 소스코드의 특징을 n-gram 사이즈 별로 통계를 낸 후에 소스 코드 안에서 N-gram의 사이즈로 구분한 특징들의 거리를 비교하여 저자 분류한다. Aylin Caliskan-Islam 등[2]은 N-gram만으로는 소스코드의 특징을 충분히 나타낼 수 없기 때문에, 소스코드를 파싱하여 AST로 만든 후에 특징을 찾았다. 그 후 AST에서 파생된 저자의 스타일을 반영하는 어휘, 구조, 그리고 통사론적(syntactic) 형태로부터 구분 가능한 목록[2]을 모아 프로그램 언어를 결정 트리로 만들어진 임의 숲(Random Forest)[9]을 통해 저자를 분류하였다.

수집하거나 변환한 특징을 이용하여 저자를 구분하는 분류기로는 임의숲같은 결정 트리와, 심층신경망 그리고 거리 비교를 통한 클러스터링 등이 있다. Ivan Krsul 등[10]의 연구에서는 소스코드 작성자의 특성을 MIT Lincoln Laboratory에서 개발한 LnkNet을 통하여 분류하였다. N-gram을 사용한 Georgia Frantzeskou 등[8]의 연구에서는 소스 코드 저자의 특징을 찾기 위해 바이트 수준으로 나누어진 N-gram 프로파일로 SCAP(Source Code Author Profiles) 접근 방식을 사용한 후 각각의 N-gram의 거리를 비교하여 저자를 판별했다. Lili Mou 등[6]의 연구에서는 AST를 트리 형태로 만들어 특징을 뽑아내는 Tree-Base Convolution Neural Network (TBCNN)를 사용했다. 이 TBCNN은 AST를 “three-way pooling”로 표현하여 “binary continuous tree”라는 2진 트리 형식으로 모델을 구성하여 학습한 후에 판별하였다. XInyu Yang 등[11]의 연구에서는 프로그램의 특징을 찾아 심층신경망 모델을 사용하는 것은 같지만 입자군집 최적화(Particle Swarm Optimization) 알고리즘을 사용하여 특징파티클의 위치를 조정하여 역전파(Back-propagation)의 매개 변수로 삼아 최적화 시간을 줄였다. Yoon Kim[12]의 연구에서는 컨볼루션 층(Convolution Layer)를 3개를 중첩시켜 컨볼루션 결합 층(Concatenated Layer)을 만들었다. Fig. 1에서 컨볼루션 층이 따로 학습을 진행한 후 출력 값을 합한 모습을 보여주고 있다. 3개의 컨볼루션 층은 각각의 다른 사이즈를 가진 필터(filter)로 합성곱을 계산한다. 다른 필터를 사용하여 합성곱을 한 후, 활성화 함수로 진행된 값들을 계산하고 결합하여 학습을 진행하게 된다. 다른 컨볼루션 층에서 나온 값을 결합함으로써 다른 필터에 따른 결과를 모두 보존하는 앙상블 모델으로 Bootstrap aggregating(Bagging)[13]처럼 병렬로 학습을 진행한다. 입력 값을 동일하게 하는 대신에 차이점을 주는 곳은 학습을 할 때의 필터로, 각각의 컨볼루션 층에서 크기를 각기 다르게 적용하여 학습을 한다. 그 후 각기 다르게 학습된 컨볼루션 층에서 나온 값들을 결합한 후에 완전 연결층(Fully Connected Layer)과 소프트맥스(softmax)를 이용하여 학습된 결과를 산출한다. 문장에 구조를 N-gram으로 변경하여 특징을 찾지 않고 학습 도중에 특징 필터의 크기를 다르게 학습 하고 과적합(overfitting)을 피하도록 하여서 분류기의 성능을 높이고 있다.

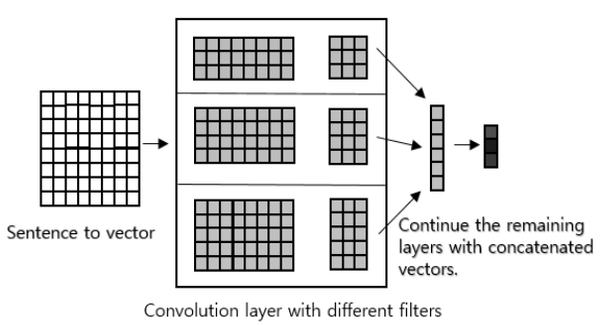


Fig. 1. Concatenated Convolution Layer and Other Layer

## 3. 전처리기와 심층신경망 구조

본 논문에서 사용한 데이터들을 바탕으로 학습하기 위한 합성 모델을 제시하고 구현하였다. 소스코드를 바탕으로 하는 저자 판별은 데이터 수집 및 정규화를 통한 벡터 변환 임베딩 층, 컨볼루션 층, 그 후 나머지 부분에서는 완전 연결 층와 소프트맥스를 이용한 저자의 클래스 판별로 볼 수 있다. 먼저 소스코드로는 저자마다 잘못된 지표를 입력할 수 있기 때문에 정확한 키워드들을 판별하려면 정규화가 필요하다. 그래서 각 키워드마다 구분이 가능하도록 불필요하거나 부정확한 문자를 변환하기 쉽도록 정규화 코드로 통일성을 맞추어준다. 각각의 단어를 잘못 인식하지 않도록 동일한 단어를 다른 표현으로 쓸 경우 동일하게 인식하도록 워드를 변경시켜 가독가능성을 높이도록 한다. 그 후 정규화를 거친 소스코드들을 벡터로 표현해 줄수 있도록 전처리기를 활용한다.

Fig. 2에서는 벡터들이 키워드들의 인덱스를 0 또는 1로 표시하도록 하는 one-hot 벡터로 표현된 것이다. 가장 기본적인 표현 방법인 이 one-hot 벡터로 소스코드들의 각각 문장을 벡터로 바꾸어 준 후에, 전처리기를 사용하여 프로그래밍 언어로써 의미를 가지고 있는 부호나 단어들을 키워드들을 좀 더 학습하기 쉽도록 입력 값을 변경 시켜준다. 예측 기반 벡터[15]의 경우에서는 한 단어의 문맥을 이해하기 위해 단어들을 가중치 학습을 하면서 매트릭스로 매핑한다. 주파수 기반 벡터[3]의 경우에는 각각의 단어들에 대한 빈도수를 통해 단어에 가중치를 부여하여 매핑한다. 이 전처리기를 통한 워드 임베딩(Word Embedding)을 통해 각각의 벡터가 단어에 대한 유사성을 내포 할 수 있도록 하고, 의미론적 관계 및 사용된 다양한 유형의 문맥을 포착 하는 단어 표현을 만들어, 컴퓨터가 이해하고 처리할 수 있도록 텍스트의 수치 표현을 사용하여 학습하기 위한 바탕을 마련한다.

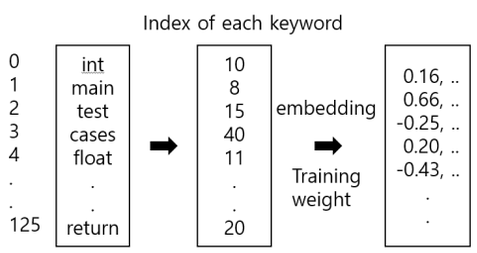


Fig. 2. Expansion Via Embedding Layer

### 3.1 예측 기반 벡터

심층학습(Deep Learning) 모델은 글자 자체를 인식하고 처리 할 수 없기 때문에, 작업을 수행하기 위한 입력 값을 변환해야하는 전처리기를 사용한다. 여기서 예측 기반 벡터를 사용하기 위한 벡터를 만들 때 모든 문자나 단어에 대한 전처리기를 사용한다면, 그 문자나 단어 값에 대한 분포가 넓게 되어 있기 때문에 학습을 효과적으로 하기 위해서 키워드 사전을 만들어 벡터를 제한한다. 그 후 임베딩 층을 통해 습 한다. 임베딩 층은 단어를 변환시켜 줄 때 학습을 하면서 동적 필터에 따라 입력 값을 변환시켜 주기 위한 계층이다[15]. 그래서 임베딩 층은 전처리기에서 변경한 벡터를 압축하거나 확장시켜 주기 위하여 사용된다. Fig. 3에서 소스코드가 각 단어가 키워드 사전을 통한 인덱싱 후에 각각의 가중치를 통한 합성곱으로 변환된 벡터는 신경심층망에서 학습하기 위해 확장된 예제이다. 이 때 확장이나 압축되는 크기는 가중치 필터의 크기로 합성곱을 하면서 결정한다. 인덱싱 값으로 구성된 벡터는 각 단어의 정보가 충분하게 분포되어 있지 않기 때문에 신경심층망으로 학습하기 힘든 벡터로 구성되어 있다. 그래서 벡터를 매트릭스로 차원을 넓힌다. 그 과정에서는 키워드를 가중치를 사용하여 매핑하게 되는데, N개의 단어들이 모여있는 확률론적 언어 모형인 N-gram으로 형성해서 연속적인 단어를 유사한 지역으로 매핑되는 모델을 만든다. 그럼으로써 이웃한 키워드를 구성하는 벡터를 만든다. 키워드 사전을 통한 인덱싱으로 문장을 벡터로 변환했을 때, 단어와 매핑되는 각각의 행을 가중치를 학습하면서 벡터를 확장되는 차원수를 곱한 매트릭스로 바꾸는 확장을 실행한다. 임베딩 층은 가중치를 이용하여 가장 적절한 매트릭스로 변환시키는데 이 때, 확장할 차원수를 정하여 변환시켜준다.

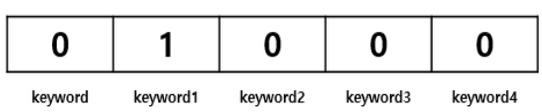


Fig. 3. It is a One-Hot Vector for Using Several Keywords as Keyword Dictionaries

### 3.2 TF-IDF

TF-IDF는 소스코드의 각 단어나 키워드들의 빈도수를 가중치로 변환하여 사용한다. 문서를 숫자 벡터로 변하는 가장 기본적인 방법은 Bag of Words(BOW)로써, BOW 방법에서는 전체 문서를 구성하는 고정된 단어장을 만들고 d라는 개별 문서에 단어장에 해당하는 단어들이 포함되어 있는지를 표시하는 방법이다. TF*(t,d)*는 단어 빈도(Term Frequency)로써 실제로 사용할 때는 찾는 단어 $t$가 개별문서 $d$에 존재하면 1, 존재하지 않는다면 0으로 환산되는 불린 빈도(Boolean Frequencies)를 사용했다, IDF*(t,d)*는 역문서 빈도(Inverse Document Frequency)로써 하나의 키워드가 문서 전체에서 얼마나 사용되는지 나타내는 값인데, 전체 문서의 수 $D$를 해당 단어 $t$를 포함한 문서 $d$의 수로 나눈뒤 로그를 취한다.

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{1}$$

$$\mathrm{tf-idf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D) \tag{2}$$

이 TF와 IDF 변환을 이용하여 신경망에서 학습시키기 위한 벡터를 만든다. 이 때 TF-IDF는 문서 전체에 대한 값을 고려하고 변환되기 때문에 전체적인 특징을 가지고 있는 값

이고, TF값과 IDF값을 곱함으로 사용된다. 이 정보를 이용하여 각각의 단어들을 숫자로 치환하여 문장들을 벡터로 형성한다. 그렇게 전체적인 단어와 문장정보들을 가지고 포함한 벡터로 학습을 하기 위한 입력 값을 만드는데, 이 때 가중치 크기에 따른 파라미터를 학습에 이용하는 단어를 제안하여 벡터의 크기를 줄일 수 있다.

### 3.3 CNN

CNN은 소스코드들의 대한 특징을 컨볼루션 층을 이용하여 학습하기 위한 심층신경망이다[16]. CNN은 문장의 지역정보를 보존함으로써 단어와 구조, 표현의 등장순서를 학습에 반영하는 아키텍처로써 모든 소스코드들이 가진 특징표현(Feature Representation Learning)을 찾아 학습한다. 전처리기를 통과한 문장 매트릭스에 이 필터를 사용하여 합성곱을 수행하고 특징 맵을 생성한다. 본 논문에서 사용한 모델은 2개의 컨볼루션 층을 가지고 있다. 예측기반벡터를 기반으로 하여 학습도중에 만들어지는 입력 값과 TF-IDF를 사용하여 만들어진 입력 값을 사용하여 학습을 진행한다. 컨볼루션 층에서 합성 곱을 이용하여 학습한 후에 완전 연결 층와 소프트맥스를 이용하고, 그라운드 트루스(Ground Truth)로 사용된 소스코드 들의 원저자들과 비교하면서 계산한다.

$$n_{out} = [\frac{n_{in} + 2p - f}{s}] + 1 \quad (3)$$

그 중 컨볼루션 층에서 가중치 필터의 행을 1로 잡아 실행한다. Equation (3)은 컨볼루션 층에서 특징 지도(Feature Map)의 크기를 정하는 수식이다. 이 수식의 하이퍼파라미터(Hyperparameter)를 보면, $n_{in}$은 컨볼루션의 입력 특징(Input Feature)값이고, $n_{out}$ 컨볼루션 층에서 필터가 입력 값과 합성곱을 통해 계산되어 나오는 값이다. 컨볼루션 층에서 f는 필터 값을 의미하고, p는 패딩(padding), s는 스트라이드(strid)로써 필터가 이동하는 값을 의미한다. 주어진 모델에서는 위의 수식 중 f, 필터의 행을 1로 정하고 학습을 수행했다. TF-IDF로 변환한 값은 각각의 문장을 매트릭스로 변환한 것이기 때문에, 행간의 상관계수는 유사도가 부족하다. 그래서 이미지 형식의 필터 형식으로 학습하기에는 부적절하다. 그래서 컨볼루션 층에서 필터 매트릭스는 행벡터로 구성하였다.

### 3.4 모델링

Fig. 4와 Fig. 5는 TF-IDF를 사용한 전치리기와 임베딩 층을 사용한 모델의 전반적인 모습을 표현한 것이다. 언어의 구별에 상관없이 두가지 전처리 방법을 통한 벡터가 CNN 레이어에서 합습하는 모델이다. 학습 모델이 CNN인 경우에 레이어를 변경해본 결과, 학습 모델 보다는 전처리기의 방법에 따라 학습 결과에 더 큰 영향을 끼쳤다. 그래서 예측기반벡터의 word2vec같은 임베딩 레이어와 TF-IDF를 사용한 워드벡터를 다르게 사용하여 학습 결과를 실험한다. 이 때 언어의 키워드들을 형태학적으로 조금씩 다르게 사용되기 때문에, 정

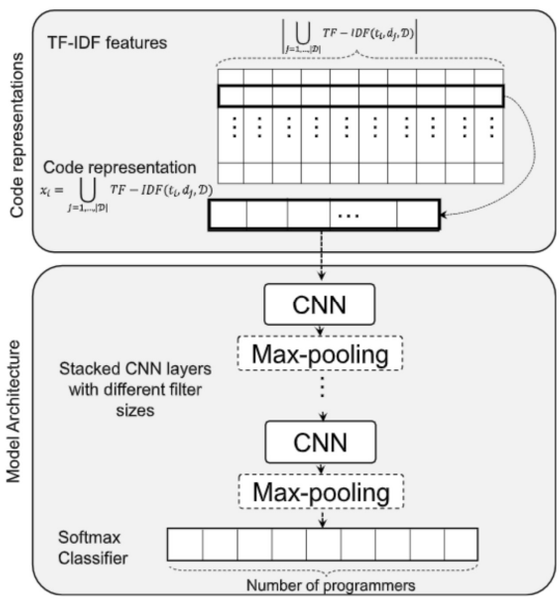


Fig. 4. The Overall Model Structure with TF-IDF

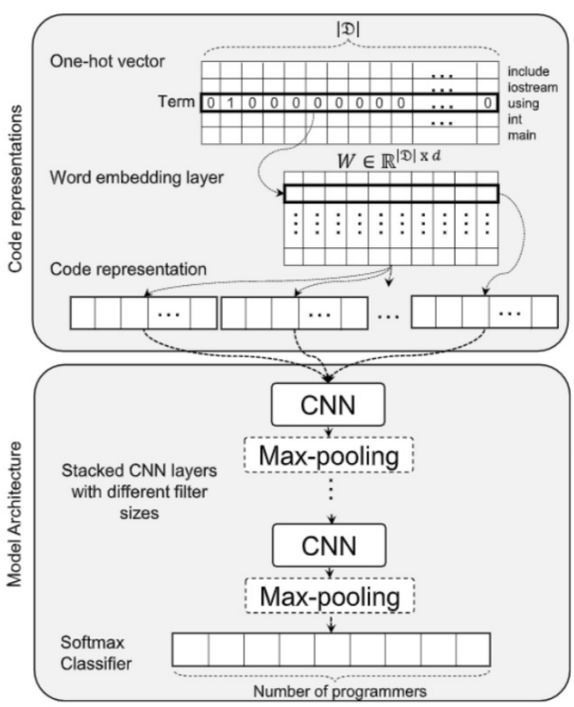


Fig. 5. The Overall Model Structure with Word Embedding Layer

규화하여 동일한 단어들로 인식하여 학습한다. 희소한 키워드들은 분포(Distribution Hypothesis) 기반으로 단어의 가중치가 배분되기 때문에 다소 과소적합한 것으로 판단되어진다. 그래서 각 키워드들의 맵핑할 시에 적절한 키워드들을 표현하여 Fig. 4로 표현되었듯이 TF-IDF를 사용하여 소스코드를 변환해준 후에 임베딩과 CNN을 다른 사이즈를 사용하여 특징을 추출한 다음 프로그래머들을 분별해주었다.

## 4. 실험 및 평가

구글 Code Jam 소소코드 중에서 C++, Java, Python의 언어들을 데이터 세트로 사용하여 Linux환경에서 Tensorflow를 사용하여 python으로 시뮬레이션을 실행하였다. 소스코드들은 100줄 이하의 소스코드들이고, 데이터 세트는 500명, 1,000명, 최대 1,500명의 프로그래머들이 9개씩 소스코드를 가지고 있다. C++의 경우에는 각각 총 4,500, 9,000, 14,500개의 소스코드들을 가지고 학습하였고, Java는 최대 1297명의 저자들로 총 11,673개의 소스코드로 실험하였다. Python은 최대 1,500명의 저자와 14,500개의 소스코드를 사용하였다. 모델은 키워드 사전과 임베딩 층을 거친 컨볼루션 결합 모델, TF-IDF를 통한 CNN을 사용하였다. 데이터의 경우 균형적으로 각각의 저자마다 동일한 개수의 소스코드를 가지고 있기 때문에 정밀도(Precision)와 재현율(Recall)은 거의 동일한 결과를 가지고 있다.

키워드 사전은 모든 키워드에 대한 값을 임베딩 층으로 확장한다면 과적합(overfitting) 하여 정확도가 떨어질 수 있기 때문에 7,000개의 사이즈로 키워드사전을 제한하였다. 임베딩 층의 차원수는 128값을 정하였다. 그리고 CNN모델의 컨볼루션 층에서 필터의 크기는 3개의 컨볼루션 층에서 3, 4, 5 또는 3, 5, 7로 사이즈를 조정하였고, CNN의 각각의 필터의 개수는 128개를 사용하였다. 학습률(learning rate)는 0.002로 실행하여 실험에서는 각각의 배치 사이즈를 64개씩 학습을 하였으며, 학습 에폭(epoch)은 100으로 맞추었다.

Table 1. Experiment with Predction Based Vector

| Language | Number of Programmer | Accuracy |
|---|---|---|
| C++ | 500 | 94.5 |
| | 1,000 | 91 |
| | 1,500 | 88 |
| Java | 500 | 93 |
| | 700 | 90.5 |
| | 1,000 | 88 |
| Python | 500 | 84 |
| | 1,000 | 76.5 |
| | 1,500 | 72 |

Table 1의 실험은 C++ 과 Java, Python 의 언어로 구성된 데이터 세트의 저자를 구분한 실험의 결과로 학습한 모델에서 테스트 데이터를 바탕으로 실행한 정확도를 나타냈다. 키워드 사전의 경우는 데이터 세트가 많아질수록 커지는데, 이때 중요한 키워드와 중요하지 않은 키워드를 구분할 수 없기 때문에 데이터 세트가 많아질수록 정확도가 떨어지는 것을 볼 수 있다. 또한 C++와 Java의 키워드는 일반적으로 많이 쓰이는 구문이 많기 때문에 Python에 비하여 학습이 잘 되었다. 그러나 Python의 경우에는 구문이 다른 언어에 비해 자유롭기 때문에 저자의 특징을 원활히 찾지 못하여 정확도가 떨어졌다.

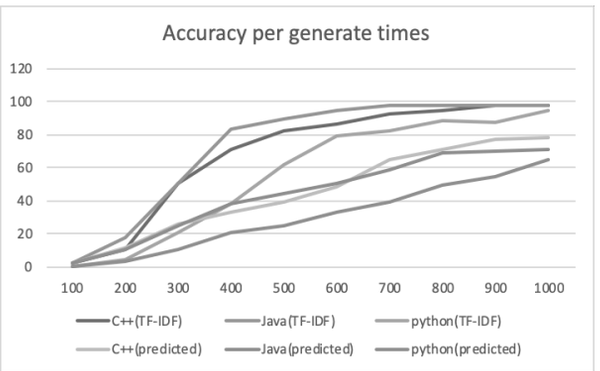


Fig. 6. Accuracy Each Language and Preprocessing Per Batch Generate Times

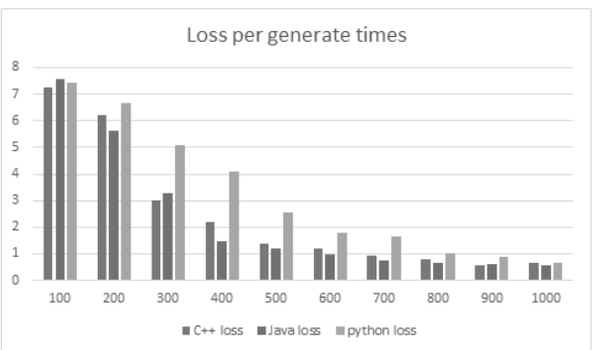


Fig. 7. Loss of Each Language Value Per Batch Generate Times in TF-IDF and CNN Model

다음은 본 논문의 모델인 TF-IDF를 전처리기로 사용하여 각각 언어의 모든 소스코드에 대한 값들을 매트릭스로 변환하여 CNN으로 학습한 그래프이다. 여기서 CNN의 경우에는 2개의 컨볼루션 층 중에서 첫 번째의 특징 필터 개수는 36개로 잡고, 두 번째 컨볼루션 층은 62개로 형성한 후에 배치 사이즈를 100으로 한후 학습을 진행시켰다.

Fig. 6은 TF-IDF와 예측기반벡터를 사용한 전처리기와 CNN으로 학습한 모델에서 테스트 세트로 실험한 프로그래밍 언어당 1000번의 배치 사이즈에서 학습한 정확도이고, Fig. 7은 그라운드 트루스와 학습한 결과의 오차를 나타낸 손실(loss) 값이다. 이때 C++은 1,500명 저자, Java는 1,297명의 저자, 그리고 Python은 1,500명 저자로 실험하였다. TF-IDF는 전체 문서대비 각 키워드의 빈도를 나타내서 키워드 중에서 중요한 키워드를 구분한다. 그래서 데이터 세트를 최대로 한 후 실험을 하여도 정확도가 빨리 올라가는 것을 볼 수 있다. 또한 Table 1과 비교를 하였을 때 더 높은 정확도를 보여주고 있다. 그 중에서 Python 데이터 세트는 여전히 다른 언어에 비해 특징이 두드러지지 않아서 느리게 학습이 진행되었다. 확인하였듯이 전처리기와 임베딩 층을 통한 입력 값 변환하는 것보다 TF-IDF를 통한 CNN을 사용하는 것이 소스코드를 이용하여 분류하는 부분에서 적은 학습량으로 더 빨리 학습하고, 조금 더 높은 정확도를 보이고 있다. 손실 값 또한 배치사이즈를 통한 학습이 1,000번이 되기전에 충분히 학습가능한 것을 보여준다.

## 5. 결 론

본 논문에서는 문장의 특성인 키워드를 보다 잘 표현할 수 있는 TF-IDF를 이용하여 매트릭스로 변형하고, 가중치를 심층신경망을 사용하여 학습하는 저자 분류 방법을 제안하였다. 소스코드 키워드에 따른 전처리기와 심층신경망이 저자를 얼마나 잘 분류 하는지 실험하였는데, TF-IDF와 심층신경망을 사용하고 있는 모델이 임베딩과 컨볼루션 층을 결합한 모델이나 그 외의 모델과 비교하여 손실 값이 더 빠르게 감소하고, 정확도 또한 더 높은 것을 보였다. 그럼으로써 더 원활히 학습하고 있다는 것을 확인했다. 한편 본 논문에서는 키워드 특징을 명확히 가지고 있는 소스코드를 데이터 세트로 활용하였고, 그럼에 따라 키워드가 명확하게 구분되는 방법을 사용하여 학습하였기 때문에, 상용어를 사용할 때도 키워드를 이용한 단어 빈도가 명확하게 특징을 찾을 수 있을지에 대해 실험하는 것을 향후 과제로 계획하고 있다.

## References


[1] S. Narayanan and S. Simi, “Source code plagiarism detection and performance analysis using fingerprint based distance measure method,” *2012 7th International Conference on Computer Science & Education (ICCSE)*. IEEE, 2012.

[2] A. Caliskan-Islam, R. Harang, A. Liu, A. Narayanan, C. Voss, F. Yamaguchi, and R. Greenstadt, “De-anonymizing programmers via code stylometry,” *24th USENIX Security Symposium (USENIX Security)*, Washington, DC. 2015.

[3] T. Joachims, “A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization,” No. CMU-CS-96-118. Carnegie-mellon univ pittsburgh pa dept of Computer Science, 1996.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word repre-sentations in vector space,” arXiv preprint arXiv:1301.3781, 2013.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, Vol.86, No.11, pp.2278-2324, 1998.

[6] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin, “Convolutional Neural Networks over Tree Structures for Programming Language Processing,” In *Thirtieth AAAI Conference on Artificial Intelligence(AAAI)*, Vol.2, No.3, 2016.

[7] W. S. Choi and S. B. Kim, “N-gram Feature Selection for Text Classification Based on Symmetrical Conditional Probability and TF-IDF,” *Journal of Korean Institute of Industrial Engineers*, Vol.41, No.4, pp.381-388, 2015.

[8] G. Frantzeskou, E. Stamatatos, S. Gritzalis, and S. Katsikas, “Source code author identification based on n-gram author profiles.” In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, Springer, Boston, MA. pp.508-515, 2006.

[9] L. Breiman, “Random forests,” *Machine Learning*, Vol.45, No.1, pp.5-32, 2001.

[10] I. Krsul and E. H. Spafford, “Authorship analysis: Identifying the author of a program,” *Computers & Security*, Vol.16, No.3, pp.233-257, 1997.

[11] X. Yang, G. Xu, Q. Li, Y. Guo, and M. Zhang, “Authorship attribution of source code by using back propagation neural network based on particle swarm optimization,” *PloS one*, Vol.12, No.11, pp.e0187204, 2017.

[12] Y. Kim, “Convolutional neural networks for sentence classification,” arXiv preprint arXiv:1408.5882, 2014.

[13] L. Breiman, “Bagging predictors,” *Machine Learning*, Vol.24, No.2, pp.123-140, 1996.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word repre-sentations in vector space,” arXiv preprint arXiv:1301.3781, 2013.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” Advances in Neural Information Processing Systems, 2013.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, 2012.



**임 지 수**

https://orcid.org/0000-0002-2631-6076
e-mail : indexxlim@gmail.com
2017년 충남대학교 컴퓨터공학과(학사)
2019년 인하대학교 컴퓨터공학과(석사)
관심분야 : Machine Learning, Data Minaing, Computer Vision, Natural Language Processing



**Tamer Abuhmed**

https://orcid.org/0000-0001-9232-4843
e-mail : tamer@inha.ac.kr
2005년 가자대학교 컴퓨터공학과(학사)
2009년 인하대학교 컴퓨터공학과(학사)
2012년 인하대학교 컴퓨터공학과(박사)
2015년~현 재 인하대학교 컴퓨터공학과 교수
관심분야 : Network Security, Wireless Sensor Network, Social Network Analysis, Applied Cryptography, Intrusion Detection System