앱 인벤터로 개발한 앱의 표절 탐지 도구 설계 및 구현

신세훈, 한동준, 한원근, 박희완
한라대학교 정보통신방송공학과
e-mail:heewanpark@halla.ac.kr

Design and Implementation of a Plagiarism Detection Tool for Apps Created with the App-Inventor

Se-Hoon Shin, Dong-Jun Han, Won-Keun Han and Heewan Park
Department of Information, Communication and Broadcasting Engineering,
Halla University

요 약

앱 인벤터는 GUI 환경에서 블록 편집기를 사용하여 앱을 개발한다. 따라서 누구나 쉽게 앱 프로그래밍을 시작할 수 있다는 장점이 있다. 또한, 앱 인벤터의 공식 사이트의 gallery 공간에 공개된 수많은 공개 앱 소스(aia 파일)를 쉽게 구할 수 있기 때문에 다른 사람이 만든 앱의 소스를 그대로 가져다가 이미지만 바꿔서 자신이 만든 것처럼 앱을 공개할 수도 있다. 그러나 직접 블록 단위로 비교해보지 않고서는 표절이나 도용 여부를 판단하는 것은 쉽지 않다.

따라서 본 논문에서는 앱 인벤터로 개발한 앱들의 유사도를 자동으로 계산해주는 도구를 개발하였다. 원본 프로그램과 도용된 프로그램은 유사도가 높게 계산될 것임을 예상할 수 있기 때문에 유사도 계산 프로그램은 코드 도용을 확인하는 목적으로 활용될 수 있다. 본 논문에서 구현한 도구의 평가를 위해서 다양한 실험을 수행하였고, 실제로 유사도가 높았던 앱들이 서로 공통된 블록을 다수 포함하고 있음을 밝혀내었다. 이러한 실험결과를 바탕으로 우리가 개발한 도구가 앱 인벤터로 개발한 앱에 대해서 소스 표절이나 코드 도용을 탐지하는 목적으로 활용될 수 있을 것으로 기대한다.

1. 서론

앱 인벤터[1]는 기존의 코딩 방식과는 달리 사전 지식이 필요없고, 미리 준비되어 있는 블록들을 서로 맞추듯이 연결하여 (그림 1)과 같이 코딩을 하는 방식으로 누구나 쉽게 안드로이드 앱을 만들 수 있도록 설계되었다.

(그림 1) 골프 게임[2] 앱의 블록 일부분 발췌

앱 인벤터의 공식 사이트[3]의 gallery 공간에는 수많은 사람들이 자신이 직접 개발한 앱을 자체 소스 포맷인 aia 파일로 공개하고 있기 때문에 공개된 aia 소스를 그대로 사용하고 디자인만 수정한 후 마치 자신이 직접 개발한 앱인 것처럼 유료로 판매하는 것도 가능하다.

또한 최근 대학 뿐만 아니라 중, 고등학교에서도 소프트웨어 교육의 일환으로 앱 인벤터를 많이 다루고 있는데, 만일 학생들이 과제 제출을 할 때 다른 사람의 소스를 그대로 가져다가 약간의 수정을 하게 된다면 눈으로 봐서는 표절이나 도용 여부를 판별하기가 쉽지 않다. 이러한 현실에도 불구하고 앱 인벤터로 만든 앱에 대해서는 표절 탐지나 도용 탐지에 관련된 연구가 시도되지 않았다.

본 논문에서는 이 문제를 해결하고자 앱 인벤터로 개발한 앱들의 유사도를 측정하는 도구를 제안하고 구현하였다. 이 도구를 이용하면 사람이 직접 블록들을 비교하지 않아도 자동으로 유사도를 계산하여 표절이나 도용 여부를 판단하는데 도움을 줄 수 있을 것으로 기대한다.

2. 관련 연구

자바나 안드로이드 앱에 대한 도용탐지 및 유사도 비교 도구로 소프트웨어 버스마크[4,5]가 사용되었다. 소프트웨어 버스마크란 프로그램의 소스 코드 기반이 아닌 바이너리에 대한 직접적인 유사도 비교 방법으로 제안된 개념이다. 프로그램은 각각이 고유한 특성을 가지고 있기 때문에 유사한 프로그램을 식별하는데 사용될 수 있다. 대표적인 소프트웨어 버스마크는 Myles 제안한 k-gram 버스마크[6]이다. k-gram이란 명령어를 추출하여 연속된 k개의 시퀀스 집합을 버스마크로 사용한다. 이 방법은 실제로 구현에 사용된 코드를 버스마크로 사용하기 때문에 서로 다른 프로그램을 구별하는 능력은 뛰어나다.

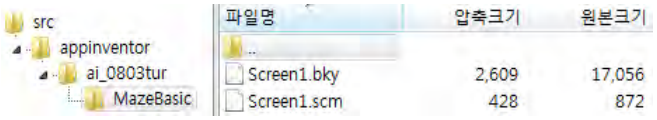
본 논문에서는 앱 인벤터의 소스에 해당하는 aia 파일로부터 블록 정보를 포함하고 있는 부분만 추출하여

k-gram 방식의 비교 알고리즘을 적용하고자 한다. 기존의 k-gram 방식은 k-gram을 생성할 때 단어 단위로 분리하였지만 본 논문에서는 앱 인벤터의 특성상 블록 단위로 프로그래밍하며 하나의 블록이 여러 라인으로 표현되기 때문에 효율성을 위해서 라인 단위를 k-gram을 생성하였다.

3. 앱 인벤터로 만든 앱들의 유사도 비교 방법

3.1 알고리즘 블록 정보(bky) 추출

앱 인벤터로 만든 앱의 알고리즘 블록에 대한 정보는 bky 파일에 저장되어 있다. aia 파일로부터 bky 파일을 추출해내기 위하여 먼저 aia파일을 zip파일로 확장자 변경을 하고 압축 해제하면 내부에 bky 파일을 찾을 수 있다.



(그림 2) aia 파일의 압축을 풀면 찾을 수 있는 bky 파일

3.2 블록 정보로부터 k-gram 생성

k-gram은 코드를 k개의 조각으로 나누어서 추출하고 비교하여 유사도를 구하는 기법이다. 예를 들어,(그림 4)와 같은 원본 블록 소스 코드가 존재한다면 먼저 유사도를 비교하는데 의미가 없는 문장들을 제거한다. 즉, 개발자에 의해 쉽게 변경될 수 있는 태그들을 의미하며, 블록 순서를 알려주는 id 값, 블록이름, 블록이 위치한 캠퍼스의 좌표, 블록이 펼쳐지는지를 알려주는 inline 값을 예로 들 수 있다. 또한 닫는 태그는 유사도를 높이는 요인이 되므로 제거한다. (그림 3)의 원본 블록으로부터 의미 없는 정보들은 (그림 4)에서 제거된 것을 확인할 수 있고, 결과적으로 (그림 5)와 같은 정보로 요약될 수 있다.

1	<block id="1" type="component_event" y="-6" x="6">
2	<mutation event_name="Initialize" instance_name="Screen1" component_type="Form"></mutation>
3	<field name="COMPONENT_SELECTOR">Screen1</field>
4	<statement name="DO">
5	<block id="2" type="component_set_get" inline="false">

(그림 3) bky 파일로부터 추출한 원본 블록 정보

1	<block id="1" type="component_event" ~~y="-6" x="6"~~>
2	<mutation event_name="Initialize" ~~instance_name="Screen1"~~ component_type="Form">~~</mutation>~~
3	<field name="COMPONENT_SELECTOR">~~Screen1</field>~~
4	<statement name="DO">
5	<block id="2" type="component_set_get" ~~inline="false"~~>

(그림 4) 유사도 비교할 때 제거되는 의미 없는 정보들

1	<block type="component_event">
2	<mutation event_name="Initialize" component_type="Form">
3	<field name="COMPONENT_SELECTOR">
4	<statement name="DO">
5	<block type="component_set_get">

(그림 5) 의미 없는 정보가 제거된 결과

1	<block type="component_event"> <mutation event_name="Initialize" component_type="Form">
2	<mutation event_name="Initialize" component_type="Form"> <field name="COMPONENT_SELECTOR">
3	<field name="COMPONENT_SELECTOR"> <statement name="DO">
⋮	⋮

(그림 6) bky 파일로부터 생성된 k-gram(k가 2인 경우)

1	<block type="component_event"> <mutation event_name="Initialize" component_type="Form"> <field name="COMPONENT_SELECTOR">
2	<mutation event_name="Initialize" component_type="Form"> <field name="COMPONENT_SELECTOR"> <statement name="DO">
3	<field name="COMPONENT_SELECTOR"> <statement name="DO"> <block type="component_set_get">
⋮	⋮

(그림 7) bky 파일로부터 생성된 k-gram(k가 3인 경우)

(그림 5)의 결과로부터 k개씩 쌍으로 묶어서 k-gram을 생성하며 k값이 2인 경우 생성된 k-gram은 (그림 6)이고, k값이 3인 경우 생성된 k-gram은 (그림 7)과 같다.

k값은 적용 대상에 따라서 변경 가능한 값이다. k가 커질수록 서로 다른 앱을 구별할 수 있는 신뢰성이 높아지는 반면, 서로 유사한 앱을 탐지할 수 있는 강인성이 낮아진다. 만약 k가 작아지면 강인성이 높아지고, 신뢰성이 낮아진다. 본 논문에서는 일반적인 도용탐지 분야에서 많이 사용되는 값인 3을 사용하였다.

3.3 k-gram 비교 및 유사도 계산

두 파일로부터 추출된 k-gram을 비교하면서 공통적으로 포함된 k-gram이 있는지 카운트 하였고, 이 결과를 이용해서 유사도 공식을 활용하면 최종 유사도를 구할 수 있다.

$$\text{유사도 공식}(\%) = \frac{n(A \cap B)}{\min(n(A), n(B))} * 100$$

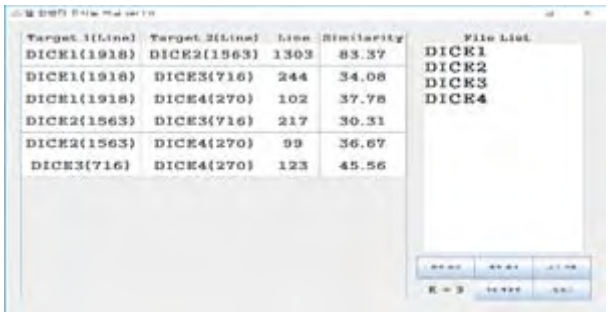
두 파일에 포함된 k-gram의 개수를 각각 n(A)와 n(B)라고 하고 두 파일에 공통 포함된 k-gram을 n(A∩B) 라고 할 때, 유사도 공식은 공통된 k의 개수를 두 프로그램

중에서 작은 프로그램으로 나눈다. 유사도 공식의 분모에서 작은 프로그램을 기준을 하는 이유는 프로그램의 크기가 서로 다를 때 어떤 프로그램이 원본인지 모르는 상황임을 전제하여 크기가 작은 프로그램을 고른 것이다.

k-gram 기법은 공통 k-gram의 개수를 이용해서 유사도 값을 계산한다. 그러나 앱 인벤터의 블록 프로그래밍 특성상 같은 블록을 사용하면 다른 프로그램일지라도 유사도가 높게 계산될 수 있으며 특히 프로그램의 매우 작은 경우 같은 블록의 사용만으로 유사도가 높게 측정 될 수 있다는 문제가 있다. 따라서 너무 작은 앱의 경우에 도용이나 표절의 의미가 없기 때문에 실험에 사용된 앱은 이러한 앱들을 배제하였다. 앱 인벤터로 생성한 앱으로부터 유사도를 계산하는 전체 과정은 (그림 8)과 같이 표현될 수 있다.

(그림 8) 유사도 비교 도구의 실행 흐름도

4. 실험 및 평가



(그림 9) 구현된 유사도 비교 도구 화면 구성

실험 및 평가를 위해서 유사도 비교 도구를 (그림 9)와 같이 자바 언어로 개발하였고, 사용 편이를 위해서 GUI 환경으로 구현하였다. 본 도구에는 bky 파일 추가, 삭제, 전체 삭제 및 k값 재설정 기능이 있으며, 왼쪽 영역에는 두 앱 사이의 유사도와 공통 k의 개수, bky 파일의 라인 수를 볼 수 있고, 오른쪽 영역에는 불러온 bky 파일 목록을 확인할 수 있다.

4.1 기존의 앱과 확장된 앱 사이의 유사도

다른 개발자가 만든 유사 앱들은 비록 유사한 기능을 수행하더라도, 서로 코드를 공유하지 않고 독자적으로 개발하였다면 공통되는 블록이 거의 존재하지 않을 것이고 앱 사이의 유사도가 낮을 것으로 예상 가능하다.

<표 1>의 실험에서는 앱 인벤터 도서[7]에 수록된 원본 앱과 유사앱(업그레이드 버전)을 대상으로 실험을 하였다. Datacal Plus 앱은 Datacal 앱의 업그레이드 버전이며, Memo Plus 앱이 Memo 앱이 업그레이드 앱이다. 예상대로 업그레이드 버전의 앱은 원본 앱과 비교했을 때 유사도가 매우 높게 나왔고 전혀 다른 앱들은 유사도가 낮게 계산되었다.

<표 1> 원본 앱과 유사앱(업그레이드 버전)의 유사도

	앱이름 (라인수)	Datacal (1041)	Datacal Plus (1054)	Memo (419)	MemoPlus (565)
Datacal (1041)	공통 k의 개수	-	1033	119	162
	유사도	-	**99.2%**	28.4%	28.6%
Datacal Plus (1054)	공통 k의 개수	-	-	123	166
	유사도	-	-	29.3%	29.3%
Memo (419)	공통 k의 개수	-	-	-	408
	유사도	-	-	-	**97.3%**
MemoPlus (565)	공통 k의 개수	-	-	-	-
	유사도	-	-	-	-

4.2 다른 개발자의 유사 앱에 대한 실험

다른 개발자가 만든 유사 앱들은 비록 유사한 기능을 수행하더라도, 서로 코드를 공유하지 않고 독자적으로 개발하였다면 공통되는 블록이 거의 존재하지 않을 것이고 앱 사이의 유사도가 낮을 것으로 예상 가능하다.

<표 2> 다른 개발자의 유사 앱에 대한 실험(1)

	앱 이름 (라인수)	Maze1 (461)	Maze2 (213)	Maze3 (278)
Maze1 (461)	공통 k의 개수	-	52	96
	유사도	-	**24.41%**	**34.53%**
Maze2 (213)	공통 k의 개수	-	-	51
	유사도	-	-	**23.94%**
Maze3 (278)	공통 k의 개수	-	-	-
	유사도	-	-	-

<표 2>의 실험에서는 앱 인벤터 갤러리[3]에서 수집한 앱 중에서 다른 개발자들이 만든 비슷한 기능의 앱을 비교했다. 그 결과 다른 개발자가 제작하였다는 것을 증명하

듯 유사도가 낮게 나왔다. 유사도가 20~30% 정도로 계산된 이유는 위에서 언급했듯이 앱 인벤터 특성상 같은 블록의 사용만으로도 공통 k-gram 개수가 증가하여 유사도가 높아질 수 있기 때문이다.

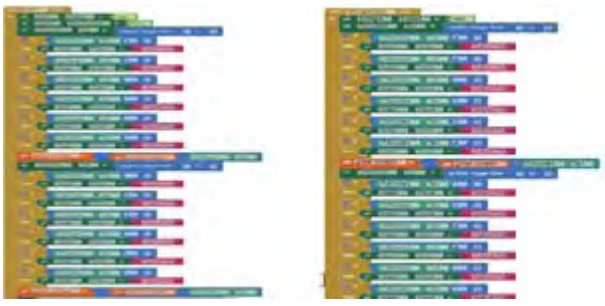
<표 3>다른 개발자의 유사 앱에 대한 실험(2)

	앱 이름 (라인수)	Dice1 (1563)	Dice2 (1918)	Dice3 (716)	Dice4 (1308)
Dice1 (1563)	공통k의 개수	-	1303	217	146
	유사도	-	83.3%	30.3%	11.1%
Dice2 (1918)	공통k의 개수	-	-	244	208
	유사도	-	-	34%	15.9%
Dice3 (716)	공통k의 개수	-	-	-	218
	유사도	-	-	-	30.4%
Dice4 (1308)	공통k의 개수	-	-	-	-
	유사도	-	-	-	-

<표 3>에서도 마찬가지로 앱 인벤터 갤러리[3]에서 수집한 앱 중에서 서로 다른 개발자가 만든 앱을 비교했는데 당연히 유사도가 낮을 것으로 예상했지만 예상과는 다르게 Dice1과 Dice2의 유사도가 높게 계산되었다.

(그림 10) Dice1과 Dice2의 화면 구성

(그림 10)에서와 같이 두 앱의 화면 구성이 다르기 때문에 서로 다른 앱이라고 생각할 수 있으나 (그림 11)에서 확인할 수 있듯이 두 앱은 서로 같은 블록을 사용하고 블록 구조가 매우 유사하다는 것을 확인할 수 있었다.



(그림 11) 실제 유사한 블록 코드

5. 결론

본 논문에서는 앱 인벤터로 만든 앱 사이의 유사도를 계산하는 방법을 제안하였다. 또한 GUI 도구를 개발하여 누구나 쉽게 이용할 수 있도록 환경을 구성하였다.

다른 개발자들이 만든 서로 다른 앱에 대해서 유사도 계산 도구를 이용한 결과, 기존의 앱과 확장된 앱에 대해서는 유사도가 높게 계산되는 것을 확인할 수 있었고, 서로 다른 앱의 경우에는 유사도가 낮게 계산된다는 것을 확인하였다. 또한 앱의 화면 디자인이 달라서 겉으로 보기에는 서로 다른 앱으로 보이는 경우에도 같은 블록을 사용하고 구조가 유사하다면 높은 유사도가 계산된다는 것을 확인할 수 있었다.

이러한 실험 결과들을 바탕으로 앱 인벤터로 개발한 앱들에 대해서 유사도를 계산하는데 편리하게 사용될 수 있음을 확인하였고, 코드 도용을 탐지하는 목적으로 활용될 수 있다는 가능성을 보여주었다.

그러나 본 논문에서 제안한 방식은 다음과 같은 한계점을 가지고 있다. 블록 프로그램을 k-gram 방식으로 비교하기 때문에 블록의 배치 순서가 바뀌거나, 한 블록 안에서도 명령어의 순서가 바뀌면 유사도가 낮아진다는 한계가 있다. 그리고 프로그램 소스의 길이가 짧으면 비교하는 대상이 적어서 몇 개의 같은 블록만 있어도 유사도가 높아지기 때문에 신뢰성이 떨어진다. 위에서 언급된 문제점들이 보완한다면 정확한 유사도 계산 및 코드 도용에 대한 탐지능력을 더욱 높일 수 있을 것으로 기대한다.

참고문헌

[1] "MIT App Inventor," http://ai2.appinventor.mit.edu.

[2] "Honey jam app inventory : Easy and interesting app programming world," J.-E. Sim, etc., Kaos Press, 2014.

[3] "MIT App Inventor Gallery," http://ai2.appinventor.mit.edu.

[4] Tamada, H., Nakamura, M., Monden, A., Matsumoto, K. Java birthmark Detecting the software theft. IEICE Transactions on Information and Systems, E88-D, 9 (Sept. 2005), 2148-2158.

[5] S.-H. Choi, H.-Y. Lee, S.-M. Cho, H.-W. Park, "API Similarity Comparison Tool Development for Detecting Theft of Android Application," Proceedings of the 37th conference of the KIPS, vol.19, no.1, pp. 792-795, Apr. 2012.

[6] Ginger Myles and Christian Collberg. k-gram Based Software Birthmarks. In Proceeding of the 2005 ACM Symposium on Applied Computing, pp. 314-318. Santa Fe, New Mexico, USA, 2005.

[7] "Challenge your app inventory! Android App Developer", Codable, Sung-An-Dang Press, 2016.