효율적인 안드로이드 앱 저장소의 개발

김연어[1], 천준석[1], 우균[2]
[1,2]부산대학교 전기전자컴퓨터공학과
[2]LG전자 스마트 제어 센터
e-mail:{yeoneo, jscheon, woogyun}@pusan.ac.kr

Developing an Efficient Apps Repository for Android

Yeoneo Kim[1], Junseok Cheon[1], Gyun Woo[2]
[1,2]Dept. of Electrical and Computer Engineering, Pusan National University
[2]Smart Control Center of LG Electronics

요 약

스마트폰 보급률이 올라감에 따라 마켓에 등록되는 앱 또한 폭발적으로 증가하고 있다. 이에 따라 앱을 대상으로 하는 다양한 연구가 진행되고 있어 연구자에게 다양한 앱을 모으는 작업이 중요해지고 있다. 이 논문에서는 안드로이드 앱을 효율적으로 저장하는 저장소를 제안하고자 한다. 제안 시스템은 임의의 앱을 입력으로 받아 자동으로 분류 후 기존 앱과 차이점만을 저장하는 것으로 용량 절감 효과가 있다. 이 논문에서는 실험을 통해 제안 시스템이 앱을 자동으로 분류함을 보였으며, 35개 앱(7종의 서로 다른 앱 5개)을 대상으로 실험한 경우 약 37~40%의 용량 절감 효과가 있는 것으로 나타났다.

1. 서론

"현대인에게 필수 불가결한 요소로 자리 잡은 물품이 무엇이 있는가?"라고 물어본다면 많은 사람들은 스마트폰을 꼽을 것이다. 스마트폰은 단순한 전화 기기에서 벗어나 메일, 인터넷 쇼핑, 온라인 뱅킹, 게임 등과 같이 다양한 영역에서 이용되고 있다. 이러한 특징 때문에 전 세계 스마트폰 보급률은 2016년을 기준으로 70%에 육박하는 것으로 나타났다. 특히 한국의 경우 스마트폰 보급률이 91%나 되는 것으로 조사되어 국민 대다수가 스마트폰을 사용하고 있는 것으로 보아도 과언이 아니다[1].

이러한 스마트폰 보급률과 맞물려 마켓에 등록된 앱도 폭발적으로 증가하고 있다. 특히 한국에서 가장 널리 사용되는 안드로이드의 경우 2015년 6월을 기준으로 160만 개가 등록되어 있으며 지금도 계속 증가하고 있다[2]. 하지만 이러한 앱의 증가는 긍정적으로만 바라볼 수 없는 것이 개인정보 유출과 같은 악의적인 행위를 발생시키는 앱도 같이 늘어나고 있기 때문이다.

이 때문에 최근에는 스마트폰 앱, 특히 안드로이드를 대상으로 하는 다양한 분석 및 연구가 활발히 이루어지고 있다. 이는 안드로이드는 마켓과 시스템이 오픈된 구조로 제3의 마켓이나 인터넷을 통해 앱을 구하여 사용자가 쉽게 설치할 수 있기 때문이다. 즉 안드로이드 앱은 마켓에 등록된 앱뿐만 아니라 다양한 경로를 통해 얻어지는 앱도 분석의 대상으로 삼아야 된다. 더욱이 앱의 버전도 고려한다면 그 수는 더욱 많아진다. 즉 안드로이드 앱을 효율적으로 수집하고 처리할 수 있는 시스템이 필요하다.

이 논문에서는 효율적으로 안드로이드 앱을 관리하는 저장소 시스템을 제안하고자 한다. 제안 시스템은 임의의 앱을 입력으로 받으면 우선 자동으로 앱을 분류하여 어떤 방식으로 저장할지를 결정한다. 제안 시스템은 앱을 분류하기 위해 표절 검사 기법[3,4]에서 사용되는 방법을 이용한다. 그리고 분류 결과 기존의 분류에 없는 새로운 앱이면 앱 자체를 저장한다. 그리고 기존 분류에 속하는, 즉 기존 앱의 다른 버전이라면 앱의 차이점만 저장한다. 이 방법을 이용한다면 단순히 앱을 저장하는 방법에 비해 저장소를 더 효율적으로 사용할 수 있다.

이 논문은 다음과 같이 구성된다. 2장에서는 관련 연구로 앱 분류 방법으로 사용되는 표절 검사 기법에 대해 살펴본다. 그리고 3장에서는 제안 시스템의 구성과 모듈별 동작 과정을 제안한다. 4장에서는 제안 시스템에 간단한 앱을 적용하는 실험을 통해 제안 시스템의 정확성을 보이고자 한다. 5장에서는 제안 시스템의 한계점에 대해 이야기하고자 한다. 이후 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

2.1 표절 검사 기법

표절 검사는 프로그램 사이의 복제 여부를 판단하는 기법이다. 표절 검사는 주로 프로그램의 언어적 특징을 추출하여 비교하는 방법이다. 그리고 표절 검사 기법은 과제 검사에서부터 라이선스 검증이나 악성코드 추출과 같은 다양한 분야에서 사용되고 있다[3]. 표절 검사 기법은 전단부와 후단부를 나누어 구성되며 전단부에서는 프로그램의 특징을 추출하며 후단부에서는 추출한 특징을 비교한다. 이러한 표절 검사 기법은 주로 소스코드나 이진(binary) 파일을 대상으로 한다.

소스코드를 대상으로 하는 대표적인 표절 검사 기법은 문자열, 토큰, AST(abstract syntax tree), PDG(program

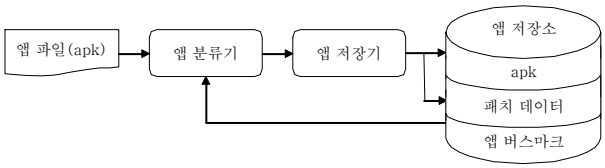
dependency graph) 등을 이용한다[3]. 문자열을 이용한 방법은 소스코드에 주석과 공백을 제거하고 일종의 텍스트 문서로 취급하여 표절 여부를 판단한다. 토큰을 이용한 방법은 어휘 분석(lexical analysis) 과정에서 발생하는 토큰 정보를 이용하는 방법이다. AST를 이용한 방법은 구문 분석(syntax analysis) 과정의 결과인 AST를 이용하는 방법이다. PDG는 소스코드에 대한 의존 그래프를 작성하고 의존 그래프 형태를 비교하여 표절 여부를 판단하는 방법이다.

이진 파일을 대상으로 하는 대표적인 방법은 버스마크(birthmark)를 이용한 방법이 있다. 버스마크는 프로그램이 가지는 고유한 특징을 이용해 표절 여부를 판단하는 방법이다[4]. 이 방법은 주로 Java를 대상으로 연구되고 있다. 버스마크는 클래스의 특징인 상수 값이나 호출 순서, 상속 관계 정보를 이용한 방법에서부터 API 호출 구조를 이용하는 방법까지 다양한 방법이 제안되었다.

이 논문에서 대상으로 하는 안드로이드 앱은 Java를 기반으로 제작되었기 때문에 역컴파일(decompile)이 쉬운 편이다. 즉, 소스코드나 이진 파일 모두 대상으로 할 수 있다. 하지만 소스코드를 대상으로 하는 방법은 대규모 프로그램에 적합하지 않다. 그러므로 이 논문에서는 이진 파일을 대상으로 하는 버스마크 기법을 이용하고자 한다.

3. 시스템 설계

이 장에서는 효율적인 안드로이드 앱을 저장할 수 있는 시스템을 제안하고자 한다. 제안 시스템은 입력으로 임의의 앱을 받아 분류한 뒤 저장소에 차이점만을 저장하는 시스템이다. 이러한 제안 시스템의 전체적인 구성은 그림 1과 같다.



(그림 1) 제안 시스템의 구조

그림 1은 제안 시스템의 구성과 제어 흐름을 나타낸 것이다. 제안 시스템은 크게 두 가지 모듈로 구성된다. 첫 번째는 안드로이드 앱 설치 파일인 apk를 입력으로 받아 앱을 분류하는 앱 분류기이다. 해당 모듈은 기존 앱 저장소에 유사한 앱이 있는지를 판단한다. 그리고 두 번째는 앱 분류기의 입력을 받아 차이점만 저장하는 앱 저장기이다. 해당 모듈은 기존에 같은 분류의 앱이 있다면 차이점을 패치 파일로 만들어 저장하고한다. 그리고 같은 분류의 앱이 없거나 차이가 크다면 apk 파일을 저장하는 모듈이다.

3.1 앱 분류기

이 절에서는 제안 시스템의 모듈 중 하나인 앱 분류기의 설계 방법을 다루고자 한다. 앱 분류기는 입력받은 앱이 기존 앱의 다른 버전인지를 검사하는 모듈로 표절 검사 기법을 이용하여 구현하고자 한다. 이러한 앱 분류기는 크게 두 가지 모듈로 나누어 설계된다.

첫 번째는 apk 파일을 분석 가능한 형태로 변환하는 모듈이다. apk 파일은 크게 권한을 다루는 매니페스트 파일과 프로그램 내부에서 사용되는 리소스 파일, 실제 동작 코드가 저장된 dex(dalvik executable format)로 이루어져 있다. 즉 앱을 분류하기 위해서는 앱의 동작 코드인 dex 파일을 분석해야 하는데 이는 Java의 클래스 파일과 유사한 이진 파일 형태이기 때문에 역컴파일이 필요하다.

제안 시스템에서는 dex를 분석하기 편한 형태인 smali 코드로 변환한다. smali 코드는 바이너리인 dex 파일을 사람이 이해할 수 있는 형태로 변환한 코드로 다양한 연구에서 사용되고 있다. 그렇기 때문에 입력받은 apk파일에 smali 프로젝트에서 제공하고 있는 baksmali 프로그램을 적용하여 smali 코드로 변환하는 것이 제안 시스템의 첫 번째 모듈이다.

두 번째 모듈은 변환된 smali 코드를 대상으로 표절 검사 기법을 적용하여 유사한 코드를 찾아주는 모듈이다. 이 단계에서는 클래스 단위로 분리된 smali 코드 사이의 유사도 점수를 계산한다. 이 모듈의 전단부에서는 표절 검사 기법 중 버스마크를 이용한다. 버스마크는 Tamada에 따르면 정의 1과 같다[4].

정의 1. (버스마크) 어떤 두 프로그램 p, q가 주어졌을 때 프로그램에서 고유한 특징을 추출하는 함수 f가 존재한다. 그리고 아래 조건을 만족하는 함수 $f(p)$가 프로그램 p의 버스마크이다.

- $f(p)$는 유일하게 p에서 얻을 수 있어야 한다.
- q가 p의 복제 프로그램이라면 $f(p)=f(q)$이다.

버스마크는 정의 1에서 첫 번째 조건과 같이 프로그램의 독립적인 특징을 얻어낼 수 있어야 한다. 그리고 두 번째 조건과 같이 복제된 프로그램의 경우 원본과 동일한 버스마크를 얻을 수 있어야 한다.

이 논문에서는 다양한 버스마크 중 단순화된 SMC(SSMC: simple sequence of method calls)를 이용하고자 한다. SMC는 Tamada에 의해 제안된 버스마크로 잘 알려진 메소드의 호출을 특징으로 사용한다[4]. 그리고 제안 시스템에서 사용하는 SSMC는 정의 2와 같다.

정의 2. (SSMC 버스마크) 임의의 앱 p로부터 디컴파일된 파일 p'이 존재하며 p'은 smali 코드의 나열 $d_1,d_2,...,d_n$으로 이루어져 있다. 이때 SSMC를 추출하는 함수 f는 invoke 계열의 코드만 가져오는 함수이다

SSMC 버스마크는 정의 2와 같이 여러 smali 코드 중

invoke 계열의 코드 나열만을 사용하는 버스마크이다. 이 방법은 기존의 SMC에서 잘 알려진 메소드 호출만을 사용하는 방법과 달리 모든 메소드 호출을 특징으로 이용한다. 그리고 후단부에서는 추출된 SSMC를 대상으로 지역정렬(local alignment) 알고리즘을 적용하여 유사도 점수를 계산한다.

지역정렬 알고리즘은 유전공학 분야에서 DNA 비교를 위해 사용되고 있는 알고리즘으로 표절 검사 분야에서도 사용된 바 있다. 그리고 지역정렬 알고리즘을 이용해 유사도를 백분율로 표기하는 방법은 식 1과 같다.

$$SIM_{AB} = \frac{LA(A,B)}{\min(LA(A,A), LA(B,B))} \qquad (식\ 1)$$

식 1은 정규화된 값을 통해 유사도 점수를 백분율로 나타내기 위해 사용된다. 식 1에서 사용된 LA함수는 지역정렬 알고리즘을 의미한다. 식 1의 과정을 거치게 되면 각 smali 코드가 몇 퍼센트 유사한지가 나오게 된다. 그리고 전체 앱에 대한 유사도는 일종의 최적 배정문제와 같아서 헝가리안 알고리즘(hungarian algorithm)을 적용하여 구한다.

3.2 앱 저장기

앱 저장기는 앱 분류기를 통해 전달된 결과를 이용해 앱을 apk파일로 저장할지 아니면 패치 파일만 저장할지를 결정하게 된다. 앱 분류기의 결과가 임계값(논문에서는 80%)미만으로 판단되면 새로운 앱으로 판단하고 apk를 저장한다. 그리고 임계값 이상이면 차이점만을 저장하기 위해 패치 파일을 생성한다. apk의 경우 이진 파일이기 때문에 이진 파일용으로 제안된 bsdiff를 이용해 패치를 생성한다[5].

4. 실험

이 장에서는 제안 시스템의 유용성 및 성능을 보이기 위해 간단한 앱을 대상으로 실험을 진행한다. 이를 위해 앱 분류기와 앱 저장기의 성능을 별도로 측정한다. 그리고 실험에 사용한 앱 파일은 온라인을 통해 apk를 다운받을 수 있는 APKPure를 이용하였다[6].

4.1 앱 분류기 성능 평가

제안 시스템에서 앱 분류기는 Java 8을 통해 구현되었다. 앱 분류기의 실험은 유사한 행동을 하는 두 가지 앱의 두 버전, 즉 총 4개의 앱을 대상으로 한다. 앱 분류기가 정상적으로 동작한다면 같은 앱의 두 버전은 정상적으로 분류되고 다른 앱은 유사도 점수가 낮게 나올 것이다. 그리고 앱 분류기의 실험으로 사용할 프로그램은 아스트로 파일 관리자(v.4.6.2.4, v.4.6.3.4)와 ES 파일 탐색기(v.4.1.6 v.4.1.4.2)이다. 그리고 앱 분류기에서 같은 앱으로 판단하는 기준인 임계 값은 80%로 설정하였다. 4개의 앱을 분류한 결과는 표 1과 같이 측정되었다.

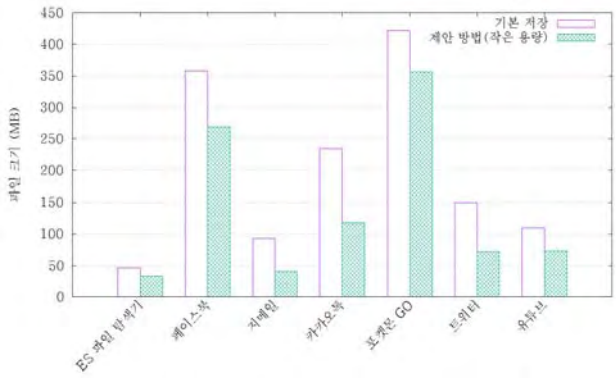
(표 1) 앱 분류기 실행 결과(유사도 단위: %)

	es_v1	es_v2	astro_v1	astro_v2
es_v1	-	95.00	32.63	32.81
es_v2	95.00	-	32.55	32.73
astro_v1	32.63	32.55	-	85.36
astro_v2	32.81	32.73	85.36	-

표 1은 아스트로 파일 관리자와 ES 파일 탐색기를 대상으로 앱 분류기를 실행한 결과이다. 실험 결과 ES 파일 탐색기 간의 유사도는 95%로 나타났으며 아스트로 파일 관리자 간의 유사도는 약 85%로 나타났다. 즉, 같은 앱 사이의 다른 버전은 분류가 정상적으로 이루어지고 있다. 그리고 ES 파일 탐색기와 아스트로 파일 관리자 사이의 유사도는 약 32%대로 나타나 서로 다른 앱임을 확인할 수 있다.

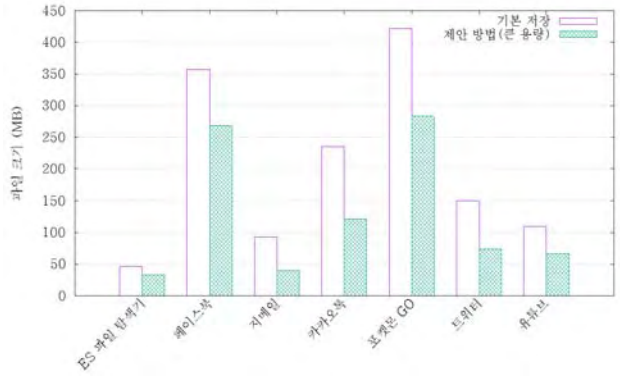
4.2 앱 저장기 성능 평가

제안 시스템의 앱 저장기는 쉘 스크립트를 통해 작성되었으며 패치 생성은 bsdiff(v.4.3-15)를 사용하였다. 이 절에서는 앱 저장기의 성능을 패치 파일로 저장하여 발생하는 용량 절감 효과를 측정한다. 그리고 패치 생성의 기반이 되는 앱 버전에 따른 패치 파일의 크기를 비교하고자 한다. 이를 위해 사용된 앱은 ES 파일 탐색기, 페이스북, 지메일, 카카오톡, 포켓몬 GO, 트위터, 유튜브와 같은 7가지 앱 중 최신 5개 버전을 사용하여 총 35개 앱을 대상으로 실험을 진행하였다. 그 결과 작은 용량을 기반으로 패치를 제작한 경우는 그림 2와 같이 측정되었다.



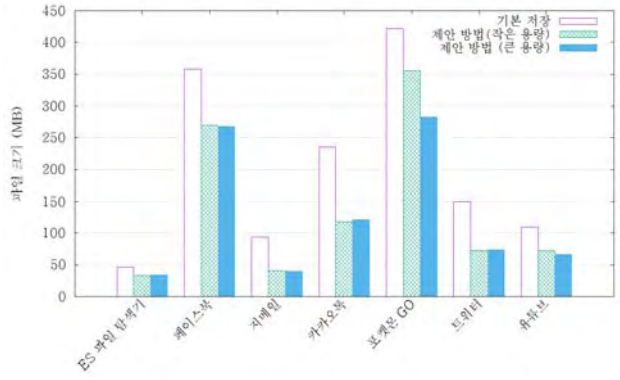
(그림 2) 작은 용량의 앱을 기반으로 패치를 제작한 경우

그림 2는 작은 용량의 앱을 기반으로 패치를 제작한 경우 측정된 저장소의 용량이다. 실험 결과 단순히 apk를 저장한 경우 저장소의 총사용량은 1,414.7MB로 나타났으며 제안 방법을 사용한 결과는 총 961.7MB를 사용하였다. 그리고 큰 용량을 기반 버전으로 사용한 경우는 그림 3과 같이 나타났다.



(그림 3) 큰 용량의 앱을 기반으로 패치를 제작한 경우

그림 3은 큰 용량의 앱을 기반으로 패치를 제작한 경우 측정된 저장소의 용량이다. 단순히 apk를 저장한 경우 앞선 실험과 같이 1,414.7MB의 공간을 사용하였으며, 제안 방법의 경우 886MB의 공간을 사용한 것으로 측정되었다. 그리고 두 실험 결과를 비교한 결과는 그림 4와 같다.



(그림 4) 패치 제작에 사용된 기반 앱 차이에 따른 성능

그림 4는 패치 제작에 사용된 기반 앱을 다르게 함으로써 얻어진 결과를 비교한 것이다. 작은 용량의 기반 앱을 사용한 경우 약 37%의 용량 절감 효과가 나타났다. 그리고 큰 용량의 기반 앱을 사용한 경우 약 40%의 용량 절감 효과가 나타났다.

5. 토의

이 장에서는 앞선 장에서 실험한 내용에 대해 논의해보고자 한다. 앱 분류기의 경우 분류 결과는 만족스럽게 나타났다. 하지만 일반적으로 앱의 규모가 커짐에 따라 생성되는 smali 코드의 숫자도 같이 증가하게 된다. 즉 지역정렬 알고리즘을 적용해야 하는 클래스의 수도 같이 증가하게 되어 실행 시간이 늘어나게 되는 문제점이 있다. 이러한 문제점을 해결하기 위해서는 일정 수 이상의 smali 코드를 생성하는 앱의 경우 다른 버스마크 적용을 고려해볼 필요가 있다.

점진적 앱 저장기의 경우 패치 제작에 사용되는 기반 앱을 선택해야 하는 문제가 있다. 실험 결과 큰 용량을 기반으로 패치를 제작하는 것이 용량 절감 효과가 뛰어나다. 그리고 일반적으로 최신 버전의 앱이 용량이 큰 편이다. 하지만 항상 최신 버전의 앱을 유지하기 위해서는 앱이 갱신되면 다른 버전의 앱에 대해 모두 패치를 다시 제작해야 하는 문제점이 있다. 즉 기반 앱을 결정하는 정책에는 장단점이 존재한다. 특히 작은 용량의 앱의 경우 특정 앱에서 용량 감소 효과가 적기 때문에 이러한 앱에 대한 분석이 필요하다.

6. 결론

이 논문에서는 다양한 버전의 안드로이드 앱을 효율적으로 저장하는 저장소 시스템을 제안하였다. 이 시스템은 임의의 앱을 입력으로 받아 자동 분류를 통해 앱 사이의 차이점만 패치 파일로 저장하는 방법이다. 그리고 실험을 통해 제안 시스템이 앱 분류를 정상적으로 함을 보였으며 35개의 앱을 기준으로 제안 시스템을 사용 시 37~40%의 용량 절감 효과가 있음을 보였다.

향후 연구로는 앱 입력 자동화를 들 수 있다. 현재 시스템은 앱을 수동으로 입력해야 하는 한계점이 있는데 이를 개선할 예정이다. 개선 방안으로는 온라인을 통해 apk를 자동으로 수집하는 웹 크롤러를 시스템에 추가하여 apk DB를 만들어주는 방법을 생각해 볼 수 있다. 그리고 또 다른 향후 연구로 고찰에서 다루었던 앱 분류기나 앱 저장기의 문제점을 분석하여 개선하는 것을 생각해 볼 수 있다.

Acknowldgement

이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2016R1D1A1B03936453).

*교신 저자 : 우균(부산대학교, woogyun@pusan.ac.kr).

참고문헌

[1] 박세정, 세계 스마트폰 보급률 70% 육박… 세계1위 한국은 몇%?[Internet],http://www.dt.co.kr/contents.html?article_no=2016070102100151780001, 2016.

[2] 유선실, 모바일 앱(App) 마켓 최근 동향, 정보통신방송정책, 제28권 3호, pp. 18-23, 2016.

[3] C. Roy and J. Cordy, “A Survey on Software Clone Detection Research,” Technical Report 541, Queen’s University at Kingston. 2007.

[4] H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto, “Java Birthmark - Detecting the Software Theft,” IEICE Transactions on Information and Systems, Vol.88, No.9, pp.2148-2158, 2005.

[5] Colin Percival, Naive differences of executable code[Internet], http://www.daemonology.net/bsdiff/, 2003.

[6] APKPure, APKPure.com: Download APK free online downloader[Internet], https://apkpure.com, 2017.