안드로이드 스마트기기를 위한 동적 애플리케이션 추천 시스템

공영선*, 하상호*
*순청향대학교 컴퓨터공학과

e-mail:vengeful224@gmail.com

A Dynamic Application Recommendation System for Android Smart Devices

Young-Sun Kong*, Sang-Ho Ha*
*Dept of Computer Science and Engineering, Soon Chun Hyang University

요 약

국내 스마트폰 보급의 폭발적인 증가와 더불어 수많은 애플리케이션이 개발되어 안드로이드 마켓 등을 통해서 제공되고 있다. 사용자의 스마트 기기 상에 많은 애플리케이션이 쌓여지고, 이에 따라서 특정 애플리케이션을 찾는데 많은 번거로움과 시간이 소요된다. 스마트 기기 상에서 애플리케이션 관리를 위한 기존 애플리케이션은 현재 사용 중이지 않으나 활성화되어 있는 애플리케이션을 삭제하거나 단순히 이름, 날짜, 혹은 이용 빈도에 기준하여 애플리케이션을 정렬하거나 추천하는 수준이다. 그러나 일반적으로 사용자에 따라서 특정 요일과 시간에 이용하는 애플리케이션이 있다. 논문에서는 요일, 시각, 시간, 이용 빈도를 고려한 애플리케이션의 사용 가중치 기반으로 동적으로 애플리케이션을 추천하는 시스템을 개발한다. 추천된 애플리케이션은 스마트 기기 시작 화면에 배치하여 애플리케이션에 대한 이용자 접근성을 향상시킨다.

1. 서론

ICT 통계월보[1]에서 보듯이 국내 스마트폰 보급의 폭발적인 증가와 더불어 수많은 애플리케이션이 개발되어 안드로이드 마켓 등을 통해서 제공되고 있다. 많은 수의 애플리케이션이 Google Play Store 등에 등록되어 있고 하루에도 수많은 애플리케이션들이 등록되고 있다. 이와 더불어서 사용자의 스마트 기기 상에 많은 애플리케이션이 쌓여지고, 이에 따라서 특정 애플리케이션을 찾는데 많은 번거로움과 시간이 소요된다.

Advanced Task Manager Pro[2], Dodol 어플 매니저[3], 어플매니저[4]와 같이 스마트 기기상의 애플리케이션을 관리해주는 애플리케이션이 나와 있다. Advanced Task Manager Pro은 현재 사용하지 않으나 활성화되어 있는 애플리케이션을 삭제하여 메모리를 관리해주는 애플리케이션이며, Dodol 어플 매니저는 이름, 날짜 그리고 크기 순서로 애플리케이션을 정렬하는 기능을 제공한다. 어플매니저는 단순히 애플리케이션의 실행 빈도수에 기반하여 애플리케이션을 정렬하여 보여주는데, 이러한 기능을 이용하기 위해서는 해당 애플리케이션을 실행시켜야하는 번거로움이 따른다. 또한, 이용자에 따라서 특정 요일과 시간에만 이용하는 애플리케이션이 존재할 수 있는데, 이러한 사항을 고려하지 않는다. 가령, 매주 금요일마다 기차를 이용하는 사람의 경우 해당 요일에 코레일 톡과 같은 기차 예약을 위한 애플리케이션을 이용하게 된다. 단순히 실행 빈도수 관점에서 애플리케이션을 추천하게 되면, 이러한 애플리케이션은 순위에서 밀리게 되어 추천되지 않게 된다.

논문에서는 애플리케이션 이용 빈도수 측정 시에 애플리케이션 이용 요일과 시간, 그리고 애플리케이션 이용 시간까지 고려한 애플리케이션 추천 시스템을 개발한다. 애플리케이션 이용 시간을 반영한 이유는 오랜 시간동안 이용하는 애플리케이션에 추천 우선순위를 주기 위함이다. 또한, 추천된 애플리케이션은 안드로이드 위젯과 알림바(Notification)를 이용하여 스마트기기 초기 화면에 배치하여 이용자 편의성을 향상시킨다.

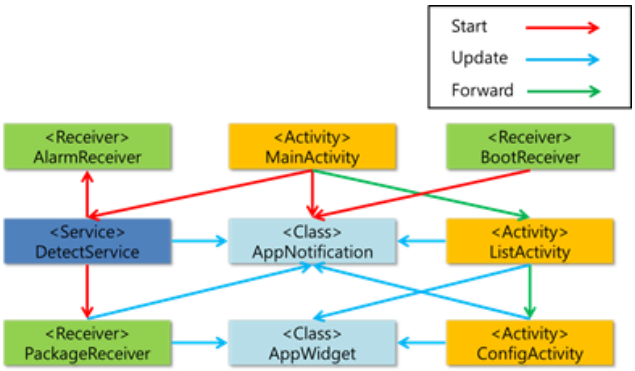
2장에서는 시스템 설계에 대해서 설명하고, 3장에서는 시스템의 구현 사항에 대해 설명한다. 4장에서는 애플리케이션 실행 예를 보여주고, 마지막으로 5장에서 결론을 언급한다.

2. 시스템

그림 1은 시스템 구성 모듈 별 관계도를 보여준다. 모듈에는 Receiver와 Activity와 Service 또는 Class로 표현된다. Receiver는 브로드캐스트 리시버를 나타내고 Activity는 화면 액티비티, Class는 자바 클래스 파일을 나타내고 Service는 서비스 동작을 나타낸다. AlarmReceiver는 최근 실행한 애플리케이션에 더 큰 가중치를 주기 위해 매주 전체 시간 별 애플리케이션 가중치

를 감소시키기 위해 애플리케이션 최초 실행 이후부터 일주일 간격마다 알려주는 리시버이다. MainActivity는 애플리케이션 실행 첫 화면이고 DetectService 실행 및 애플리케이션 설정 확인을 한다. BootReceiver는 디바이스 부팅 시 DetectService를 시작하기 위한 리시버이다. DetectService는 실행 중인 애플리케이션을 5초 주기로 현재 화면에서 실행되는 애플리케이션을 감지하여 애플리케이션의 시간 별 가중치를 실행 빈도와 실행 시간을 기준으로 계산하여 저장한다. 또한 30분 주기로 안드로이드 알림바를 갱신하고 리시버를 등록한다. AppNotification은 안드로이드 알림바 생성 및 알림바 갱신하기 위한 클래스이다. ListActivity는 애플리케이션 별 가중치를 시간과 요일 별로 볼 수 있고, 체크 박스를 이용하여 바로가기에 제외시킬 수 있는 인터페이스를 제공한다. PackageReceiver는 디바이스에서 애플리케이션 설치나 삭제를 감지하는 리시버로 설치나 삭제 시 위젯과 알림바를 갱신한다. AppWidget은 안드로이드 위젯 생성과 갱신을 위한 클래스이다. ConfigActivity는 시스템 설정 및 인터페이스 설정을 위한 액티비티이다.

MainActivity에서 DetectService를 호출하고 설정에 따라 알림바를 생성한다. 그리고 ListActivity를 호출하고 MainActivity는 종료한다. ListActivity에서는 메뉴 버튼을 통해 ConfigActivity에 접근할 수 있다.



(그림 1) 시스템 구성 모듈

시스템에는 App과 Blacklist 2개의 데이터베이스 테이블이 있다. App 테이블은 애플리케이션의 요일과 시간 별 애플리케이션 실행 횟수와 실행 횟수에 비례한 가중치를 나타내는 테이블이다. App 테이블의 칼럼에는 애플리케이션의 패키지를 나타내는 Package, 요일을 나타내는 Day, 시간을 나타내는 Hour, 실행 횟수를 나타내는 Count, 가중치를 나타내는 Weight이 있다.

Blacklist 바로가기로 보여주지 않을 어플리케이션을 저장하는 테이블이다. ListActivity 화면에서 체크박스를 통해 등록하고 해제할 수 있다. 패키지명과 블랙리스트 여부를 나타내는 칼럼으로 이루어져 있다.

애플리케이션 정렬을 위한 우선순위 알고리즘으로는 첫 번째 규칙은 요일과 시간의 일치, 두 번째 규칙으로는 시간, 세 번째로는 같은 요일에서 가중치 높은 순으로 정렬한다.

그림 2에서는 가중치 산출 알고리즘의 의사코드를 나타낸다. w는 다음번에 저장될 가중치를 나타낸다. w에 초기 값을 준다. 논문에서는 1로 가정하였다. 가중치를 구하기 위해서 실행되는 애플리케이션이 감지되면 현재 요일과 시간을 구해 실행할 애플리케이션의 가중치에 w값을 더한다. w는 실행 시간이 길어질수록 증가한다. 더 정밀한 가중치를 부여하기 위해 현재 시간에는 가중치 계수를 그대로 부여하고 1시간 전후에는 가중치 계수의 $\frac{1}{2}$만큼 그리고 2시간 전후에는 가중치 계수의 $\frac{1}{4}$만큼 부여한다. 식으로 표현하면 다음과 같다.

$$w_{pdh-2} \leftarrow w_{pdh-2} + \frac{w}{4},\ w_{pdh-1} \leftarrow w_{pdh-1} + \frac{w}{2}$$

$$w_{pdh} \leftarrow w_{pdh} + w$$

$$w_{pdh+1} \leftarrow w_{pdh+1} + \frac{w}{2},\ w_{pdh+2} \leftarrow w_{pdh+2} + \frac{w}{4}$$

$$(w_{pdh} = p\text{애플리케이션}\ d\text{요일}\ h\text{시간가중치}$$
$$w = \text{이후저장될가중치})$$

같은 애플리케이션의 실행이 반복 감지되어도 가중치를 부여하지만 실행 시간보다 실행 빈도수에 우선순위를 부여하기 위해 다음과 같은 식을 적용하여 w의 증가 폭을 감소시키고 가중치를 계산한다.

$$w \leftarrow w \cdot k\ (0 < k < 1)$$

k는 사용자 계수로 0과 1 사이의 값을 가진다. 논문에서는 $k = \frac{3}{4}$로 가정하였다. 그리고 이전과 다른 애플리케이션이 실행 될 경우 가중치 계수를 초기화한다.

```
While( DetectService is Running ) do
        d <- 현재요일, h <- 현재시간정보
        beforeApp <- app
        app <- Running Application on Screen
        If( app != beforeApp) Then
                Init w // 가중치 계수 1로 초기화
        If( app = beforeApp) Then
                w←w • k
        End If
        w_pdh ← w_pdh + w
        w_pdh ← w_pdh ± 1 + w/2
        w_pdh ← w_pdh ± 2 + w/4
        Sleep 5 second
End While
```

(그림 2) 가중치 산출 알고리즘

또한 최근 실행한 애플리케이션에 과거 실행한 애플리케이션보다 더 많은 가중치를 주기 위해 다음과 같은 식을 적용한다.

$$w_{dt} = \sum_{i=1}^{n} k'^{n-i} w_i \ (w_i = i\text{번째 주 가중치})$$

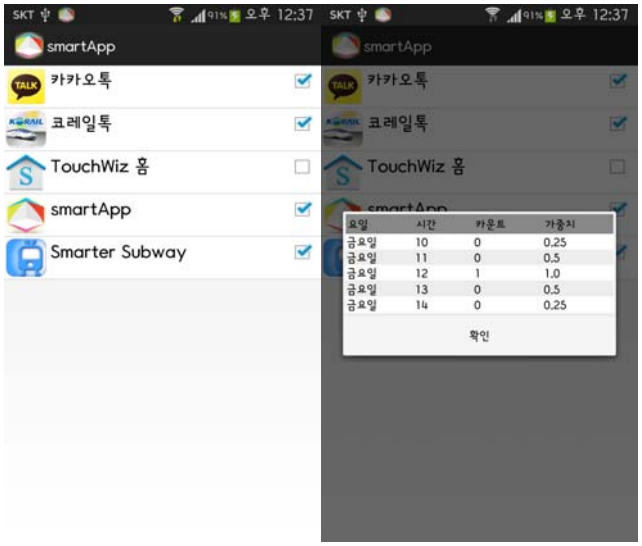
논문에서 $k' = \frac{3}{4}$로 가정한다.

3.구현

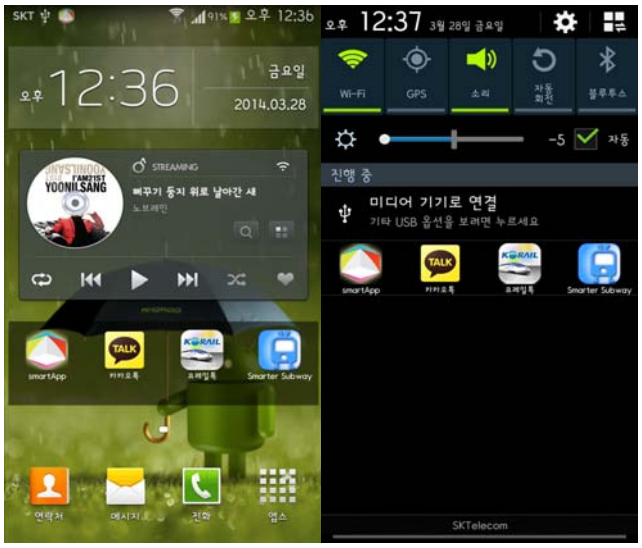
애플리케이션 개발은 Android 4.1.2 버전을 기준으로 개발하였고 데이터베이스는 SQLite 2.0 버전을 사용하였다. 개발 도구로는 통합 개발 환경 도구인 Eclipse를 사용하였다. 설정이나 애플리케이션 최초 실행 시간 같은 간단한 데이터는 안드로이드에서 제공하는 SharedPreference에 저장하였다. 테스트는 엑시노스 4412가 탑재된 Odrioid U2에 HDMI LCD모니터를 연결하여 테스트하였고 스마트폰으로는 엑시노스 4412가 탑재된 Galaxy S3를 이용하였다.

4. 실행

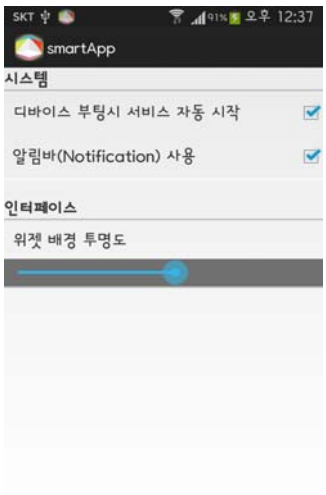
그림 3은 시스템 실행 예를 보여준다. 처음 애플리케이션을 시작하면 애플리케이션은 MainActivity가 실행된다. MainActivity에서는 첫 애플리케이션 시작이면 주기를 위한 현재 시간을 저장한다. 그리고 DetectService가 실행중이지 않으면 DetectService를 시작하고 (a)가 나타내는 ListActivity를 화면에 출력하고 MainActivity는 종료한다. ListActivity에는 리스트뷰 방식으로 지금까지 실행했던 애플리케이션들이 아이콘과 이름으로 나온다. 오른쪽에 보이는 체크박스를 통해 해당 애플리케이션을 블랙리스트 테이블에 등록할 수 있다. (b) 화면은 리스트뷰를 클릭하면 다이얼로그 안에 App 데이터베이스를 테이블 형식으로 보여주는 화면이다. (c)와 같이 안드로이드 위젯의 경우 1 * 4의 크기를 가진다. 위젯에 애플리케이션의 아이콘과 이름이 나타나고 아이콘을 클릭하면 해당 애플리케이션이 실행된다. (d)와 같이 알림바(Notification)의 경우 4개의 애플리케이션이 아이콘과 이름으로 나타난다. 애플리케이션의 아이콘을 클릭하면 해당 애플리케이션이 바로 실행된다. (e)과 같이 안드로이드 메뉴 버튼을 눌러 설정 화면으로 갈 수 있다. 설정 화면에서 설정 할 수 있는 메뉴는 디바이스 부팅 시 서비스 자동 시작 여부와 알림바(Notification) 사용 여부를 체크할 수 있다. 또한 인터페이스 설정 부분으로는 위젯 배경 투명도를 SeekBar를 통해 설정 할 수 있고 SeekBar의 배경색으로 바로 확인 할 수 있다.



(a) (b)



(c) (d)



(e)

(그림 3) 시스템 실행 예

5.결 론

논문에서는 안드로이드 스마트기기를 위한 동적 애플리케이션 추천 시스템을 개발하였다. 안드로이드에서 애플리케이션 실행은 브로드캐스트를 보내지 않기 때문에 애플리케이션 실행을 체크하기 위해 백그라운드 서비스를 이용하여 측정하였다. 그에 따라 추가적인 배터리 소모와 리소스를 필요하다. 백그라운드에서 주기적으로 체크하는 시간을 길게 하면 정확한 측정이 힘들어지고 시간을 짧게 하면 많은 리소스와 전력 등의 자원이 필요하다. 따라서 많은 테스트를 하여 애플리케이션 실행 감지 주기를 구하는 것이 필요하다. 또한 애플리케이션 실행 시 사용되는 가중치 계수의 정확성을 더 많은 실험을 통해서 높일 수 있다.

참고문헌

[1] http://www.nipa.kr/

[2] http://hwsecter.tistory.com/727

[3] http://dodol.com/

[4] http://iphone.co.kr/index.php?document_srl=67167