ユーザの状況を考慮したアプリケーション推薦システムの構築

2012SE197 岡本裕貴　2012SE198 岡村知典
指導教員：河野浩之

1　はじめに

スマートフォンは様々な用途に応じたアプリケーションをインストールすることで，ユーザの好みに合わせたカスタマイズをすることができる．また，スマートフォンのユーザ全体の 5 割ほどがアプリケーションを 10 個以上を利用しており，30 個を越えるユーザも 1 割程度だが存在する [1]．しかし，現状ではユーザが目的に応じてインストールされたアプリケーションの中から手動で選択，起動しなければならない．先行研究では TF-IDF の概念を用いてコンテキストに依存したアプリケーション推薦法によって端末の操作を簡易化する研究 [2] が行われている．そこで我々は，ユーザに合わせたアプリケーション推薦システムを実装することで，操作の手間を省き，スマートフォンの利便性の向上を目的とする．

本研究では，時間や場所などのユーザを取り巻く環境を特徴量として用いた押川の研究 [3] や，使用している端末の状態を考慮した嶋谷らの手法 [4] を用いて，ユーザが求めているアプリケーションを推薦する iOS 向けのシステムを構築する．ユーザの状況に合わせたアプリケーションを推薦することで操作が簡略化され，目的に応じたアプリケーションの選択を早くすることができる．

本研究の構成を以下に示す．2 章では，スマートフォン利用者のコンテキストログを用いたアプリケーション推薦システム，スマートフォンの使用履歴に基づいた個人モデリングの先行研究について述べる．3 章では，アプリケーションの推薦システムの流れを紹介する．4 章では収集したデータを用いた Weka による決定木モデル作成 [5] と，Xcode を用いた iOS 向けアプリケーションの開発について述べる．5 章では，推薦システムの評価を行い，6 章では本研究のむすびについて述べる．

2　アプリケーション推薦に関する先行研究

2.1　スマートフォンの使用履歴に基づいた個人モデリング [3]

押川は利用者の位置情報，時間情報，移動情報，アプリケーション履歴を関連付けることにより，ユーザ個人の使用アプリケーション推定を行った．使用アプリケーションの情報及び移動状態の情報はユーザ自身が判断し，スマートフォンを用いて取得した．そして取得した情報を元にアプリケーション使用特徴の抽出を行った．使用特徴を元に可能性の点数付けを行い，その点数を元に特定のコンテキストでのアプリケーション使用の順位付けを行うことで，そのコンテキストでのユーザの使用するアプリケーションの推定をした．結果として，全アプリケーションでの平均順位が上昇した．元々使用頻度が高いアプリケーションは順位が下がる傾向にあり，コンテキストに依存したアプリケーションの推薦精度は向上した．

2.2　スマートフォン利用者のコンテキストログを用いたアプリケーション推薦システム [4]

嶋谷らは利用者の状況や利用履歴をもとに，利用者のコンテキストを推定し，求められているアプリケーションを推定する手法を提案した．提案手法では，特徴量と特徴量の関係をグラフネットワークと考え，クラスタリングによってコンテキストの抽出を行い，各アプリケーションに対応付けられているコンテキストのスコアを求め，スコア順に推薦を行った．提案手法の有用性について検証するために，従来手法の利用回数が多い物が上位に並ぶ MFU 法，最近使用した物ほど上位に並ぶ MRU 法との比較実験を行った．結果として，提案手法は MFU 法，MRU 法よりは良いと言えないが，一部のアプリケーションでは推薦精度が良いことがわかった．

2.3　推薦システムの課題

押川らの研究ではコンテキストとして位置情報，時間情報，移動情報，アプリケーション使用履歴を用いていたが，コンテキストへの依存度を考慮すると使用頻度が高いアプリケーションの推定順位が下がってしまうという課題があげられた．嶋谷らの研究の提案手法では，従来の推薦システムの研究で用いられてきた手法より特定のアプリケーションを除いて平均的に推薦精度劣ってしまう結果となった．

3　本研究の提案手法

3.1　先行研究の問題点の改善方法

先行研究の課題を解決する方法として利用者とアプリケーションとの関連性を持たせるために，時間，場所，センサデータなどの利用状況を特徴として抽出を行う．また，ノイズとなる特徴量が存在するためどの特徴量を用いるか選択する必要があると考えられる．そして推薦精度を高めるためには先行研究の手法とは異なるアルゴリズムを用いる必要があると考えた．そこで我々はデータマイニングツールである Weka を用いて，適切なパラメータの設定を行った上で決定木モデルの作成をした．そしてユーザの状況をもとに決定木モデルを用いた推薦を行う．

3.2　アプリケーション推薦システムの流れ

アプリケーションの推薦はユーザがダウンロードしてある端末の中から上位 4 種類のアプリケーションを推薦する．1 番上位に推薦されたアプリケーションに関しては推薦システムから直接起動できるようにする．

図 1 に推薦システムの流れを示す．(1) では現在のユーザの位置情報を緯度と経度，高度の 3 つの数値として記録する．(2) では現在の時刻を取得し，Hour の数値を記録する．(3) では端末の向きや傾きを数値として記録する．(4) では weka で生成した C4.5 の決定木モデルを用いて，収集したデータをもとに，ユーザに合わせたパターンの選択をした．パターンを選択する際には Weka のパラメータの数値を入力することで，C4.5 のアルゴリズムの詳細な設定を行った．(5) では (1) から (3) までのユーザの状況を数値として記録する機能と，(4) で構成した推薦アルゴリズムを iOS 向けのアプリケーション内に実装した．本アプリケーションは起動すると自動的にデータを収集し，状況に合わせたアプリケーションを画面上に出力する．

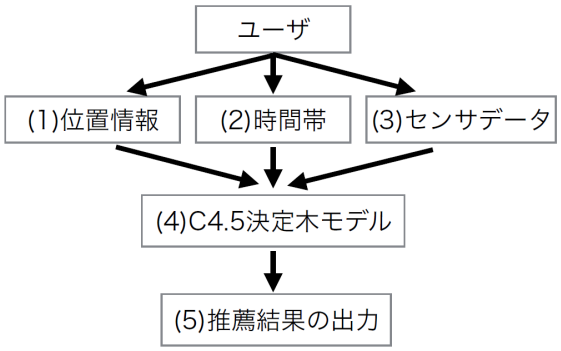

図 1　推薦システムの流れ

4　実装・実験

4.1　データ収集

ユーザの状況を取得するために，11 種類の特徴量を収集した．以下に収集したデータについて示す．

- 位置情報
  現在地の緯度，経度，高度を Geographica を用いて測定する．
- 時間帯
  アプリケーションの起動前の時間を記録する．時間を 1 時間毎の 24 個に区分した．
- センサデータ
  加速度ロガーを用いてユーザの加速度センサやジャイロセンサといった端末の向きや傾きを示した数値データを収集する．

iPhone のアプリケーションの利用状況を記録するために約 6 週間データの収集を行った．被験者 2 人はそれぞれ iPhone5s(ユーザ 1)，iPhone6(ユーザ 2) を用いた．ユーザ 1 は 17 種類のアプリケーションの起動を 305 回，ユーザ 2 は 28 種類を 472 回アプリケーションの起動を行った．アプリケーション起動回数の集計結果の一部としてユーザ 2 のアプリケーション利用状況を表 1 に示す．

表 1　アプリケーションの利用状況 (一部)

アプリケーション名	起動回数
パズル＆ドラゴンズ	75
LINE	48
⋮	⋮
Amazon	1
歌志軒 HP	1

4.2　Weka を用いた推薦アルゴリズム

アプリケーションの推薦には変数選択アルゴリズムや予測モデルの評価機能など，機械学習に関連した多くの機能を持つ Weka を用いる．Weka の分類機能を用いて事前に収集したデータを読み込み，アプリケーションの予測を行った． Weka の分類機能で J48 による交差検証を行い，アプリケーションの予測を行った．J48 は C4.5 を Weka に実装したものであり，枝刈り有無両方の C4.5 決定木を生成することができる．

図 2 に J48 のパラメータ設定をしたときの決定木の出力結果の一部を示す．決定木の分類では 472 個のデータに対して 70 種類の状況に分岐され，アプリケーション使用時の分類ができた．図の円で表されているのは属性であり，決定木の分岐を示している．四角で表されているのはクラスであり，決定木の分岐結果がそのアプリケーションの使用状況を表している．YouTube は加速度 $X \leq -0.369$ かつ加速度 $Z \leq -0.828$ かつジャイロセンサ $X \leq -1.046$ のとき 4 回使用され 1 回は誤った分類を表している．これらの結果により MUSIC，カメラ，時計，2BRO などの一部のアプリケーションでは精度の高い予測結果を出すことができた．

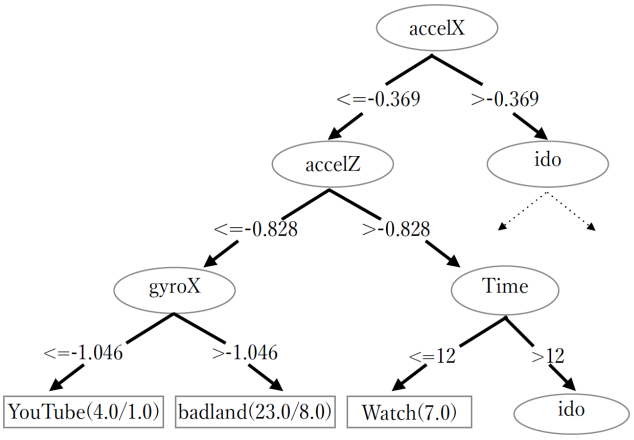

図 2　決定木 (一部)

4.3 アプリケーション開発

アプリケーションの開発には Mac に搭載されている Xcode を使用し，開発用の言語は Swift を用いる．開発したアプリケーションの実験には iOS シミュレータを利用し，位置情報やデバイスのセンサを利用した機能を試験する場合には iPhone にアプリケーションをビルドすることで，実機によるテストを行った．アプリケーションの機能としては次の 2 点を実装する．アプリケーションの開発環境を表 2 に示す．

表 2 実装環境

PC	MacBook Air	
OS	OS X v10.11.2 EI Caption	
メモリ	8GB	
CPU	1.6GHz Intel Core i5	
iPhone	5s	6
OS	iOS 9.2	iOS 9.2
メモリ	1024 MB	1024 MB
CPU	Apple A7	Apple A8

- 時間や位置情報, センサデータの取得
  各種フレームワークを利用して実装, 可視化する．
- アプリケーションの推薦システム
  Weka によって算出された決定木のモデルを利用することで実装する．

プログラムは Xcode の ViewContollor.Swift 上に記述し，各機能の構築を行い，テンプレートには Single-ViewApplication を用いて，3 つのフレームワークと搭載されたクラスを主に利用する．ユーザインターフェースは Xcode の Main.Storyboard 上で編集し，構築する．Main.Storyboard ではオブジェクトライブラリから必要な機能をドラッグ，ドロップすることで直感的にアプリケーションの画面構成を構築することができる．また info.plist から NSLocationWhenInUseUsageDescription の設定を有効にしておくことで，ユーザの位置情報をアプリケーションが利用することを任意に許可することができる．本アプリケーションでは Label にセンサデータと位置情報, 時間情報を出力し，Button には推薦結果に応じたアプリケーションへのリンク機能を搭載した．

時間情報を取得するプログラムの一部を図 3 に示す．UIKit は Xcode のフレームワークの 1 つで，アプリケーションやユーザインタフェイスの構築と管理に必要な基本的な機能が含まれている．画面の構成やプログラムとの連携は UIKit によって実現される．図 3 の 1 行目に UIKit のフレームワークの導入を行い，2 行目では UIKit に含まれてる NSDate クラスを用いることで時間情報を取得する．3 行目で，取得した情報を NSCalender クラスによって日時の数値化を行う．

```
import UIKit
let calendar = NSCalender.currentCalender()
let components = calendar.components([.Year, .Month
,.Day, .Hour, .Second, .Minute], fromDate:NSDate())
var time:Int = components.hour
```

図 3 時間情報を取得するプログラム

図 4 にセンサデータを取得するプログラムの一部を示す．CoreMotion はデバイスのセンサデータを取得するフレームワークである．CoreMotion は UIKit とは異なり初期設定として Xcode に適用されていないため，図 4 の 1 行目で Xcode のフレームワークライブラリから手動で追加している．2 行目では CoreMotion に含まれてる CMMotionManager クラスによってセンサデータの更新頻度や条件づけなどの設定が可能になる．4 行目で motionDate から各センサデータを取得可能にした．

```
import CoreMotion
let cmManager = CMMotionManager()
func motionAnimation(motionData:CMDeviceMotion?,
error:NSError?){}
let motion = motionData
```

図 4 センサデータを取得するプログラム

位置情報を取得するプログラムの一部を図 5 に示す．CoreLocation は GPS を用いた緯度や経度, 高度などの位置情報を取得するフレームワークである．図 5 の 1 行目で Xcode のフレームワークライブラリから CoreLocation を追加している．2 行目では CoreLocation に含まれる CLLocation クラスによって位置情報を取得し，3 行目で location.last から位置情報を数値として取得した．

```
import CoreLocation
var locationManager = CLLocation
let locationDate = locations.last
```

図 5 位置情報を取得するプログラム

図 6 に決定木のプログラムの一部を示す．推薦には C4.5 の決定木モデルをプログラムで記述したものを利用した．図 6 の 1 行目は決定木の分岐の条件であり，2 行目から 5 行目は分岐先のクラスでの出力結果を記載したものである．6 行目では推薦の順位が最も高いアプリケーションのリンクを生成する．

```
if gyroX <= -1.046{
one.text = "YouTube"
two.text = "BADLAND"
three.text = "Watch"
four.text = "Camera"
url = NSURL(string: "https://m.youtube.com")!}
else{
```

図 6 C4.5 の決定木モデルのプログラム

図 7 に実機にビルドし実際に起動したときのアプリケーション使用画面を示す．各クラスで取得したデータをプログラムで記述した決定木モデルに走査することで上位 4 位までアプリケーションを推薦し，順位に応じて指定した Label をアプリケーション名に変更し，推薦の順位が最も高いアプリケーションへのリンクを構築する．またユーザの状況は絶えず変化するため，5 秒おきに位置情報やセンサデータの数値から再推薦を行い，出力結果を更新するようにした．上から順に xyz 軸それぞれの加速度センサの数値，xyz 軸それぞれのジャイロセンサの数値，緯度と経度，高度といった位置情報，推薦結果を上位から 4 つ出力され，最後に現在の時刻が表示されている．

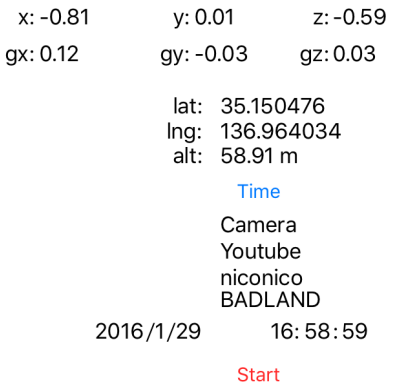

図 7 開発したアプリケーションの実機使用画面

5 推薦システムの評価

評価を行うために，構築したアプリケーションを実際に被験者 2 名の iPhone にインストールし，様々な状況下で利用した．被験者はアプリケーションや Web サービスを利用する前に本アプリケーションを使用し，推薦による出力結果と被験者が利用したいと思っていたものが一致した場合，推薦が成功したとみなした．実験は 1 人 200 回行った．表 3 は被験者 2 名のアプリケーション利用回数と本アプリケーションでの推薦結果の正誤率を示している．実験により，C4.5 の決定木モデルをアプリケーションの推薦に用いたとき，被験者 2 名とも推薦の正誤率が 70 %近くとなった．

表 3 全アプリケーション推薦の正誤率

	種類	利用回数	正誤率
ユーザ 1	17	305	0.655
ユーザ 2	28	472	0.715

表 4 は利用回数が少ないアプリケーションの利用回数と推薦結果の正誤率を示したものである．先行研究の課題となっていた，利用回数の少ないアプリケーションの推薦も，実験より比較的高い精度の推薦ができていることが確認できた．

表 4 利用回数の少ないアプリケーションの正誤率

ユーザ 2	利用回数	C4.5
YouTube	20/472	0.6667
Camera	16/472	0.8

これらの結果から位置情報やセンサデータ，時間情報といった特徴量をもとにユーザの状況に応じたアプリケーションの推薦ができているため，C4.5 の決定木モデルを用いた推薦システムの構築は概ね成功していると考えられる．

6 むすび

本研究ではユーザのアプリケーション使用時の状況を取得するために 6 週間データ収集を行い，C4.5 の決定木モデルを用いてアプリケーション使用予測を行うシステムを構築した．推薦精度については被験者 2 名の推薦の正誤率は 70 %近くになり，利用回数の少ないアプリケーションに関しても良好な結果が得られた．しかし，一部のアプリケーションでは推薦精度が著しく低下したものもあり，起動回数が極端に少ないアプリケーションに関しては Weka による予測パターンに含まれなかったため全く推薦されなかった．今後の課題としては，アプリケーションの使用履歴をもとに予測をするなど，推薦に用いる特徴量を増やしたり，多くのデータを収集した上でよりアプリケーションの使用予測精度の高いアルゴリズムの実装を行うことで精度が向上すると考えられる．また，アプリケーションの機能として決定木モデルを自動更新する学習機能を搭載することで利便性が向上すると考えられる．

参考文献

[1] 総務省:「平成 24 年スマートフォン・タブレット端末への移行とアプリ等への影響」〈http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h24/html/nc122220.html〉.

[2] 松本光弘，清原良三，沼尾正行，栗原聡，“ 携帯端末におけるコンテキスト依存アプリケーションの抽出とアプリケーション推薦法の提案 ”，情報処理学会研究報告，vol.2012-ICS-165，No.3，2012/1/12.

[3] 押川英将，“ スマートフォンの使用履歴に基づいた個人モデリング，” 法政大学大学院紀要 (情報科学研究科編)，Vol.8，pp.295-297，2013.

[4] 嶋谷健太郎，間下以大，原隆浩，清川清，竹村治雄，西尾章治郎，“ スマートフォン利用者のコンテキストログを用いたアプリケーション，” 電子情報通信学会技術研究報告，アドホックネットワーク，Vol.112，No.494，pp.101-108，2013.

[5] Anand Rajaraman，Jeffrey David Ullman，(岩野和生，浦本直彦)『大規模データのマイニング』共立出版，2014，354p.