Vol.2014-SE-186 No.4
2014/11/13

レビューサイトの情報を利用したスマートフォンアプリケーションの開発支援

清 雄一[1,a)]　田原 康之[1]　大須賀 昭彦[1]

概要：スマートフォン用のアプリケーション（スマホアプリ）開発が盛んに行われている．しかし，スマホアプリは様々なプラットフォーム，環境下で実行されるため，網羅的なテストが難しく，不具合が入り込みやすいという課題がある．さらに，ユーザの年齢やスマートフォン操作の習熟度が幅広く，ユーザの要求抽出が困難であるという課題もある．本論文では，スマホアプリのレビューサイトに投稿されているレビュー内容を分析することによって，必要とされている機能/非機能を抽出する手法及び，不具合の発生をいち早く検知する手法を提案する．

Support system for smartphone application development based on analysis of user reviews

Yuichi Sei[1,a)]　Yasuyuki Tahara[1]　Akihiko Ohsuga[1]

Abstract: A number of smartphone applications have been developed these days. However, it is difficult to develop smartphone applications without bugs because they are used in various platforms and environments. Moreover, requirements elicitation is also difficult because various persons may use the smartphone applications. In this paper, we propose an algorithm for eliciting requirements and detecting bugs early by analyzing user reviews posted in review site of smartphone applications.

1. はじめに

スマートフォンの普及に伴い，スマートフォン用のアプリケーション（スマホアプリ）の開発が盛んに行われている．スマホアプリは，Google Play や Apple Store 等で提供されているが，それぞれ登録されているアプリ数は 100 万件を超えている [1]．またスマホアプリを配布しているサイトでは，ユーザが自由にレビューを書き込めるようになっている場合が多い．人気アプリの場合は 1 日あたり数百件以上のレビューが書き込まれることもある．

このようにスマホアプリ開発やその利用は今後も拡大していくことが予想されるが，スマホアプリ開発には通常のソフトウェア開発とは異なった難しさがある．

1 つに，利用者が多岐にわたっていることである．通常のソフトウェア開発であれば，利用者や発注者と蜜に連携して要求を獲得することができるが [14]，スマホアプリ開発ではこのような方法で要求抽出を行うことは難しい．何故なら，ユーザの年齢やスマートフォン操作の習熟度，スマホアプリに対する要望が幅広く，ユーザの要求抽出が困難であると考えられるためである．また，利用する機器の性能や画面の大きさも様々であり，それらに依存した要求も多岐にわたる．

2 つ目に，スマホアプリが動作する機器や環境が多種多様であり，その全てにおいて網羅的なテストを行うことが困難であることが挙げられる．

本研究では，要望や不具合検知の手段として，スマホアプリのレビューサイトを活用することを考える．レビューには，「ここを改善して欲しい」といった機能・品質要望に関する書込みや，「正しく動作しない」といった不具合報告に関する書込み等，アプリ開発者にとって有用なものも多数含まれている．

[1] 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications, Chofu, Tokyo 182-8585, Japan
[a)] sei@is.uec.ac.jp

不具合報告については，特に人気のあるアプリでない場合は，1 日高々数十件程度のレビューであるため，開発者自身が日々閲覧することは可能であろう．しかしながら，株取引やその日のスケジュール管理を行うアプリ等，不具合が数分でも長引くと深刻な影響を及ぼす場合もあるため，できるだけ早く不具合を検知する必要がある．したがって，開発者はレビューが投稿されるたびにそれをチェックすることが望ましい．しかしながら，当然それを実施することは困難である．また，5 章に示すように，不具合に迅速に対応すると，ユーザからの評価が大きく向上する場合があることから，不具合が深刻な影響を及ぼすとまでは言えないアプリにおいても，まず可能な限り早く不具合を認識することが大切である．

機能・品質要望については，開発者自身が開発しているアプリのみではなく，同種の他のアプリに関する書込みも参考になると考えられる．この場合，関連するレビューは 1 日あたり数百件〜数千件に上る場合もあるため，開発者自身がその全てを読み込むことは現実的ではない．

そこで本論文では，以下の主に 3 つの機能から成るシステムを提案する．

- 機能/非機能の要求抽出：
  指定した単一のアプリまたは同種類の複数アプリに関するレビューから，これらアプリに期待されている機能及び非機能の要望を抽出し，開発者に提示する．
- 不具合検知：
  各レビューが不具合報告を行っているかどうかを判定し，不具合報告である場合は開発者に通知する．
- 評価の変動表示：
  指定したアプリの評価及び主なコメントの変遷を開発者に提示する．

本論文の構成を以下に示す．2 章では背景を記す．次に関連研究を 3 章に示す．提案システムを 4 章で述べる．評価を 5 章で行い，6 章で考察する．最後に 7 章で本論文をまとめる．

2. 背景

2.1 レビューの構成

Google Play や App Store 等のスマホアプリ提供サイトでは，ユーザレビューは以下に示すような要素から構成されていることが多い．図 1 は，Google Play におけるレビュー例を示している．

- 投稿者名
- 評価点
- 投稿日時
- レビュータイトル
- レビュー本文

Google Play 及び App Store では，評価点は 1〜5 までの 5 段階評価が行われている．本論文でもこのように想定する．

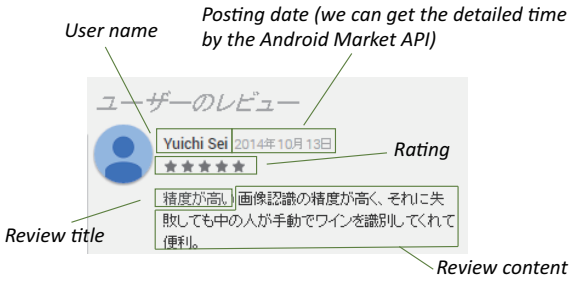


図 1 レビュー例
Fig. 1 Sample of a review.

2.2 想定シナリオ

スマホアプリ開発者が，自分のアプリ ID を指定する．指定されたスマホアプリにおける，機能/非機能の要求抽出，不具合検知及び，評価点の変動表示を行う．

機能/非機能の要求抽出については，単一のアプリ ID ではなく，複数のアプリ ID を指定することができる．たとえば，カレンダーアプリを作成している，または作成しようとしている開発者は，類似アプリ（ここではカレンダーアプリ）の ID 集合を指定することで，類似アプリに対して共通に要求されいる事項を確認することができる．開発者は，要求されている事項を確認し，そのうちのどれを自分のアプリに組み入れるかを考えることができる．

不具合検知については，開発者は自分のアプリ ID を指定するとともに，検知のレベル（偽陰性と偽陽性のトレードオフを取ることができる．詳細は 4.2 にて述べる）を指定する．また，不具合発生時の連絡先としてメールアドレスを指定する．提案システムはレビューを随時チェックし，不具合が発生した可能性が高いと判断した場合，指定されたメールアドレスにアラートを送る．アラートを受け取った開発者は，レビュー内容を確認し，不具合が発生していると判断出来る場合はすぐに対応する．

評価の変動表示については，開発者は自分のアプリ ID を指定するとともに，表示したい期間を指定する．また，特にレビューが多数投稿されている日については，代表的なレビューが併せて表示される．開発者は，評価点の変動表示を定期的に閲覧することにより，スマホアプリのアップデートの効果や，全体的に評価点が上昇傾向にあるかどうか等を確認することができる．

3. 関連研究

ユーザをソフトウェア開発に巻き込むことが，ソフトウェア開発の成功のために重要である [13]．近年では，スマホアプリユーザをターゲットとして，ユーザのフィードバックを得ることで開発に活かす研究が盛んに行われている [2]．

たとえば Seyff らは，各ユーザが，簡単にスマホアプリに

対する機能要望を提供できるようスマートフォン用のツールを作成している [11,12]．Jones らは，積極的にスマホアプリに対する機能要望を発信したいと考えるユーザが，協調的に機能を導出するためのフレームワークを提案している [9]．

特に，スマホアプリのレビューサイトを対象にマイニングを行い，アプリ開発者に情報提供を行う研究が，近年いくつか提案されている．

Fu らは，スマホアプリのレビューから，アプリが何故好かれているか，嫌われているかをマイニングする Wiscom システムを提案している [6]．Wiscom システムも本論文が提案する評価の変動表示を行うことが可能であるが，高評価または低評価のレビューが多数投稿された日（以下「ピーク」と表記する）における，代表的なレビュー抽出機能は無い．したがって，たとえば過去 1 年間の変遷を見て，ピークが複数回発生している場合，各ピークがどのような原因で発生しているのかを確認するためには，開発者が当該日それぞれのレビューを改めて確認する必要がある．また，要求抽出機能や不具合検知機能も備わっていない．

Chen らは，スマホアプリのレビューをマイニングする AR-Miner と呼ばれるシステムを提案している [4]．AR-Miner では，まず有益なレビューと有益でないレビューとを機械学習を用いて区別する．次に，本論文で提案するシステムと同じようにトピックモデルを用いてトピックを抽出している．しかし，代表的なレビュー抽出機能は無い．また，評価の変動表示機能や不具合検知機能も備わっていない．

Galvis Carreño らは，スマホアプリの要求進化のためのレビュー解析手法を提案している [7]．本論文と同じようにトピックモデルを用いてレビューを分析しているが，特に新しい要望の抽出を注視している．しかし，各トピックにおける代表的なレビュー抽出機能，評価の変動表示機能や不具合検知機能は備わっていない．

一方，スマホアプリの使用状況等をログとして蓄積し，それを分析することで要求抽出や不具合検知を試みる研究も行われている [5,16]．これらの研究と本提案手法は併用することが可能であると考えられる．

4. 提案システム

提案システムの概要を図 2 に示す．4.1 で要求抽出について述べ，4.2 において不具合検知について述べる．最後に，評価の変動表示について 4.3 で記述する．

4.1 機能/非機能の要求抽出

開発者が指定したアプリまたはアプリ群のレビューから，当該アプリ（群）に望まれることの多い機能を抽出する．アプリ群は，開発者が明示的に指定することもできるし，アプリマーケットの検索窓からキーワードで検索した結果得られたアプリ集合を利用することもできる．

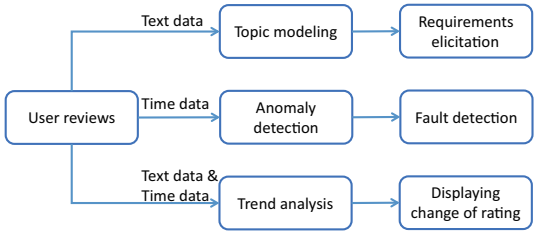


図 2 システム概要
Fig. 2 System overview.

開発者が指定したアプリに関するレビュー集合に対し，トピックモデルの一つである Latent Dirichlet Allocation (LDA) [3] を用いて，各単語を確率的にトピックに分類する．トピックモデルは，与えられた文書集合から潜在的なトピックを推定する教師無し学習モデルである．一つの文書（ここでは一つのレビュー）には複数のトピックが存在すると仮定し，各トピックが単語（ここではレビューに含まれる各単語）の分布として表現される．LDA はトピックモデルの中でも最も広く利用されているものの一つである．単語をトピックごとに分類することによって，ユーザの意見を大まかにクラスタリングすることができると考えられる．

また，各トピックへの適合度が最も高いツイートを，そのトピックの代表的なツイートとして抽出する．ただし，3 単語以上含むものに限定する．アルゴリズムを Algorithm 1 に示す．

Algorithm 1 Extracting representative reviews of each topic

Input: レビューの集合 R（指定されたアプリ ID 集合），トピック数 $topicNum$
Output: 各トピック t_i を代表するレビュー $\hat{r_i}$

```
map (Key: a pair of word and topic ID, Value: the probability that the word is categorized to the topic) ⇐ getLDA(R, topicNum)
Create array maxValues with size topicNum
Craete array repReviews with size topicNum
for r ∈ R do
    Create array temps with size topicNum
    for w ∈ getWords(r) do
        for i = 1, ..., topicNum do
            temps[i] ⇐ temps[i] + map.get((w, i))
        end for
    end for
    for i = 1, ..., topicNum do
        if temps[i]/ln(Length(r)) > maxValues[i] then
            maxValues[i] ⇐ temps[i]/ln(Length(r))
            repReviews[i] ⇐ r
        end if
    end for
end for
return repReviews
```

ここで，$map\langle(w,i),v\rangle$ は，(w,i) をキー，v を値とする

マップ関数である．機能/非機能抽出の具体例は評価結果に記載している図 1 を参照のこと．

4.2 不具合検知

各レビューは，各ユーザが任意のタイミングで投稿している．したがって，通常時においては，レビューはポアソン過程に従って投稿されると仮定することができる．これは，低評価レビュー（評価点が 1 または 2）も高評価レビュー（評価点が 4 または 5）も同じである．

しかし不具合発生時は，通常時のレビュー投稿とは異なり，多くのユーザが近いタイミングで低評価レビューを投稿する．したがって，この場合は通常時のポアソン過程とは逸脱した投稿が行われると考えられる．

提案システムでは，通常時において低評価レビューが投稿される頻度を開発者があらかじめ指定しておく．この指定された頻度から，通常時における低評価レビューが投稿されるタイミングについてのポアソン過程を導出する．このポアソン過程から逸脱し，短い間隔で低評価レビューが投稿された場合，アラートメッセージを開発者に送信する．

X をポアソン過程における事象の発生回数，λ を単位時間あたり事象の平均発生回数とすると，単位時間に r 回発生する確率は，

$$P(r,\lambda) = \frac{\lambda^r}{r!}e^{-\lambda} \tag{1}$$

と表される．λ の値は開発者があらかじめ指定しておく．

直近の期間 T 秒に対し，低評価レビュー数発生数を N_T とし，

$$V = \sum_{r=N_T}^{\infty} P(r,\lambda T) = 1 - \sum_{r=0}^{N_T-1} \frac{(\lambda T)^r}{r!}e^{-\lambda T} \tag{2}$$

を計算し，開発者があらかじめ設定した閾値 sl に対して

$$V < sl \tag{3}$$

となった場合は，この発生頻度が通常のポアソン過程を逸脱している可能性が高いと判断し，アラートを出す．

低評価レビューが発生するたびに，T を動かしながら式 2 を計算し，閾値を超えているかどうかを式 3 で確認する．

ここで，T の取り得る値を以下のように制限する．T は，少なくとも 2 件の低評価レビューが存在するよう定める．また，直近で低評価レビューが発生した時刻 t_0 からさかのぼって $t_1,\ldots,t_i,\ldots$ と数えるとき，T は，$(t_i - t_0) + (t_i - t_0)/2i$ を満たすように設定する．V の値が減少する限り i の数を増やしていく．i の数を増やして T の範囲を拡大していき，式 3 を満たす前に V の値が上昇に転じた場合は，直近に発生した低評価レビューは，ポアソン過程を逸脱して投稿されたレビューではないと判断し，アラートは出さない．逆に式 3 が満たされた場合はアラートを出す．

不具合検知アルゴリズムを Algorithm 2 に示す．ここで，$L.get(i)$ は，リスト L に対し，i 番目の要素を取り出す操作を表す．

閾値 sl の値を大きくすると偽陽性が発生する確率が低下する一方，偽陰性が発生する確率が上昇する．逆に，sl の値を小さくすると偽陽性が発生する確率が低下する一方，偽陰性が発生する確率が上昇する．

オプションとして，低評価レビューが発生するたびに開発者にアラートを出すこともできる．

Algorithm 2 Anomaly detection

Input: 低評価レビューが投稿された日時のリスト（直近のものから降順）L，通常時における単位時間あたりの低評価レビュー発生頻度 λ，閾値 sl
Output: アラートを発生させるかどうか（True または False）

```
minV ⇐ 1
t_0 ⇐ L.get(0)
for N_t = 2, ..., |L| do
    t_i ⇐ L.get(N_t − 1)
    T ⇐ (t_i − t_0) + (t_i − t_0)/(2(N_t − 1))
    V ⇐ 1.0
    for r = 0, ..., N_t − 1 do
        V ⇐ V − (λT)^r e^{−λT}/r!
    end for
    if V < sl then
        return TRUE
    end if
    if V > minV then
        return FALSE
    else
        minV ⇐ V
    end if
end for
return FALSE
```

4.3 評価の変動表示

Google Play や App Store では，各アプリに対してユーザが 5 段階評価を行うことができる．この評価の変動を可視化する．

まず，指定されたアプリ ID を基に，当該アプリ ID に関するレビューをスマホアプリ提供サイトから取得する．開発者が指定した期間内の各日 d_i において，以下の計算を行う．ここで，R_i はある 1 日 d_i において投稿されたレビュー集合を表し，$r.point$ はあるレビュー r における評価点を表す．

$$\begin{aligned} SP_i &= \sum_{r\in R_i} \max(r.point - 3, 0) \\ NP_i &= \sum_{r\in R_i} \min(r.point - 3, 0) \end{aligned} \tag{4}$$

SP_i は，高評価レビュー（評価点が 4 または 5）について，評価点 4 を+1，評価点 5 を+2 とみなし，各日 d_i において足し合わせた結果を表している．同様に，NP_i は，低評価レビュー（評価点が 1 または 2）について，評価点 2 を-1，評価点 1 を-2 とみなし，各日 d_i において足し合わせた結果を表している．

開発者に，SP_i 及び NP_i が指定された期間内においてどのように変動しているのかを提示する．

さらに，高評価及び低評価それぞれ絶対値が上位 α 日（α の値は開発者が指定する）に対して，代表的なコメントを抽出する．代表的なコメント抽出は以下のとおり行う．まず各日の各レビュー r について，高評価（または低評価）における，レビュー r を除く全レビューを形態素解析し，含まれる単語（名詞，動詞，形容詞）の集合を作成する．

次に，レビュー r が，集合に含まれる単語数を何個含んでいるかをカウントする．開発者に代表的なレビューを提示することを考慮すると，同様の内容であれば文章が短いほうが良い．そこで，カウント数を，レビューの文章の長さの対数で割る．その結果得られた値が最も高いレビューを，代表的なレビューとして抽出する．アルゴリズムを Algorithm 3 に示す．

ここで，$getWords(r)$ は，レビュー r を形態素解析して得られた単語集合を返す関数である．また，$Length(r)$ は，レビュー r の文字数を返す関数である．

Algorithm 3 Extracting a representative review

Input: レビューの集合 R（指定されたアプリ ID，日付，評価点）
Output: R を代表するレビュー $\hat{r}$

```
Create variable maxValue initialized to 0
for r1 ∈ R do
    Create empty set S
    for r2 ∈ R do
        if r2 ≠ r1 then
            S ⇐ S ∪ getWords(r2)
        end if
    end for
    Create variable c initialized to 0
    for w ∈ getWords(r1) do
        if S contains w then
            c ⇐ c + 1
        end if
    end for
    if c/ln(Length(r1)) > maxValue then
        maxValue ⇐ c/ln(Length(r1))
        r̂ ⇐ r1
    end if
end for
```

この評価の変動表示の具体例は，評価結果に掲載している図 4 を参照のこと．

5. 評価

5.1 実装

スマホアプリ提供サイトとして，Google Play を利用した．オープンソースとして公開されている android market api [*1] を利用して，Google Play から自動的にレビューを取得するプログラムを Java で実装した．

[*1] https://code.google.com/p/android-market-api/

各レビューの形態素解析については Mecab ツールを利用した．また，無視する単語（ストップワード）や辞書は適宜追加した．抽出する単語は名詞，動詞，形容詞に限定した．また，表記揺れ対策として，形態素解析後の各単語について，英文字は全て大文字に変換した．LDA の実装は，Collapsed Gibbs Sampling [8] のアルゴリズムを用いた．

5.2 評価結果

まず，機能/非機能の要求抽出について評価を行った．Google Play の検索窓に「カレンダー」と入力して検索し，検索結果の上位 50 アプリを対象とした．総計 17,588 件のレビューを取得した．

このカレンダーアプリ群について，機能要望の抽出機能の実験を行った結果を表 1 に示す．表内の「Topic」列は，任意に付与した番号であり，順序に意味は無い．「Top 5 words in each topic」列は，各トピックに対し，寄与度が高いトップ 5 つの単語を載せている．「Representative review」列は，各トピックにおける代表的なレビューを一つ掲載している．「コロン（：）」で，レビュータイトルとレビュー内容を分けている．

表より，次のような機能要求が抽出されると考えられる．シンプルなものが良い（トピック 3）．通知設定が欲しい（トピック 5）．可愛いものが良い（トピック 6，13）．Google カレンダーとの同期機能が欲しい（トピック 11）．ウィジェットの色や文字を変更する機能が欲しい（トピック 12）．日記を書く機能が欲しい（トピック 16）．誕生日を設定できる機能が欲しい（トピック 17）．広告を無くして欲しい（トピック 19）．

また，以下のような不具合報告も抽出された．データが消えてしまった（トピック 15）．ここで抽出される不具合は，このような状況が起こらないようにスマホアプリを開発すべきだという要求としてもとらえることができる．

なお，トピック 9 の上位の単語の大部分は，顔文字に利用される記号であった．顔文字専用の辞書を追加することができれば，顔文字を分析対象から除外することで，トピック 9 のようなあまり有益でないトピックが生成されないようにすることができると考えられる．

表 1 から分かるとおり，カレンダーアプリに要求されている機能及び非機能が複数個抽出できていると言える．

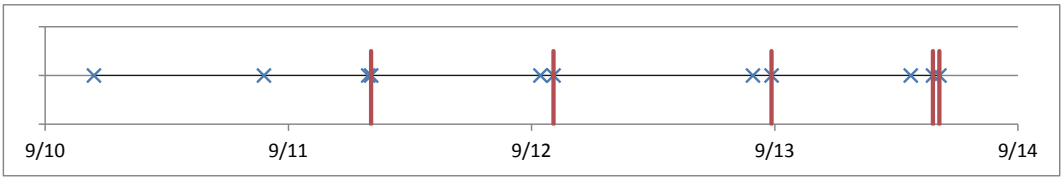
次に，不具合検知について評価を行った．Amazon アプリに対して不具合検知を行った結果を図 3 に示す．図 3 は，2014 年 9 月 10 日から 2014 年 9 月 13 日までの期間において，どのタイミングでアラートが発生したかを示すものである．図における縦棒がアラート発生のタイミングを示している．図中の×印は，参考までに，低評価レビューが投稿された日時を示している．

2014 年 3 月〜8 月の 6 ヶ月間において，Amazon アプリに対する低評価レビューの発生頻度が約 1/42727.0（回/

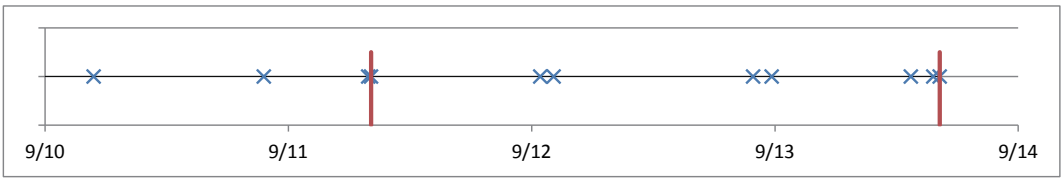
Topic	Top 5 words in each topic	Representative review
1	良い，わかる，よい，分かる，使い勝手	よい: 良いアプリ
2	アプリ，スケジュール，管理，私，今	いいです: 今まで使ってきたスケジュール管理アプリの中で 1 番ですね♪ほんと使いやすい (` ・ ω ・ ´)
3	機能，欲しい，嬉しい，シンプル，デザイン	シンプル: 欲しい機能だけ付いてます(^_^)v
4	いい，すごい，いう，感じ，わかりやすい	いい感じ: 上等
5	設定，通知，入力，時間，時	PC で…: ブラウザから予定を入力する時、通知の設定が出来ないのがイタイ
6	スタンプ，かわいい，多い，種類，簡単	☆かわいい: スタンプが便利
7	表示，日，月，カレンダー，見る	カレンダー: 今日の一日の予定が詳しく表示されるようにしてほしい
8	使いやすい，可愛い，♪，簡単，お気に入り	可愛くて: 使いやすい♪
9	(，)，O，笑，	あ: うん (笑)
10	予定，アイコン，入れる，削除，登録	予定表が便利ですよ:
11	カレンダー，同期，GOOGLE，ジョルテ，スマホ	良い: Google カレンダーとの同期は便利！
12	ウィジェット，色，文字，使い，変更	良い: シンプルで使いやすいです。週の開始曜日とウィジェットの文字色、欲を言えばフォントも変えれるなら最高。
13	使いやすい，可愛い，見やすい，(**)，分かりやすい	可愛いし見やすいし使いやすい:
14	使える，嵐，最高，無料，愛用	最高: 無料でこれだけ使えれば文句無し
15	消える，対応，アップデート，データ，機種	データ消失: いつも便利に使わして貰ってましたが本日 4.1.2 にバージョンアップするとデータが全て消えました。(以下略)
16	日記，写真，便利，書く，手帳	可愛くて予定、一言日記をつけるのが楽しい: 使いやすい！予定を入れるのが楽しくなる！日記も書けるし写真も自動で撮った日に表示してくれる (以下略)
17	便利，誕生日，忘れる，助かる，友達	友達の誕生日を: 忘れないので便利！
18	改善，画面，更新，お願い，起動	改善お願いします: 最近更新してからアプリが開けず強制終了になります。 予定が確認出来なくてこまってます。
19	広告，インストール，アプリ，何，星	新着メッセージが一件あります: 何人かの方が言ってるこの広告?に私も騙された (笑) かなり鬱陶しいですね。これがなければいいアプリだと思います。
20	使う，やすい，重宝，気に入る，もらう	使いやすく可愛い:

表 1 トピック分類結果

Table 1 Result of extracted topics



(a) $sl = 0.01$



(b) $sl = 0.05$

図 3 異常検知の結果（×印は，低評価レビューが投稿されたことを示す．縦棒は，提案システムによってアラートが発生したことを示す．）

Fig. 3 Results of anomaly detection

秒）であったため，この実験では $\lambda = 1/40000.0$ として実験を行った．閾値 sl の値を 0.01 に設定した結果を図 3(a) に，0.05 に設定した結果を図 3(b) に示している．

$sl = 0.05$ に設定した図 3(b) では，低評価レビューが連続して発生している 9/11 と 9/13 のそれぞれ 1 回ずつのみアラートが発生している．一方，$sl = 0.01$ に設定した図

3(a) では，さらに加えて，多少遅れて低評価レビューが発生している 9/12 を含め，全部で 5 回アラートが発生していることが分かる．このように，sl の値を調整することによって，アラートを発生させやすくするかどうかを調整することが可能となる．

9/10 における低評価レビューは，機能要望と，「出品者に連絡しようとするとパスワード入力画面が延々と繰り返される」といった不具合報告の内容であった．後者の内容は，筆者らが確認したところ 2011 年 5 月から 2014 年 9 月の 3 年間で 5 件だけレビューが投稿されており，提案システムでは現状では検知できない．投稿数が少ないことを考えると，アプリの不具合ではなくユーザの手違いか，または，ほとんど利用されていない機能であるため，不具合対策の優先順位が低い機能であると考えられる．このような，ごく低頻度であるが，長期にわたって報告されるような不具合や機能要望についても，将来課題として，抽出することを考えている．

9/11 における低評価レビューはアプリのアップデート後，強制終了してしまうというレビューや，使いづらくなったというレビューが投稿されている．9/12 も同様の投稿である．9/13 は，「問題が発生して買い物ができない」旨のレビューが連続して投稿されており，速やかに対応すべき不具合だと考えられる．

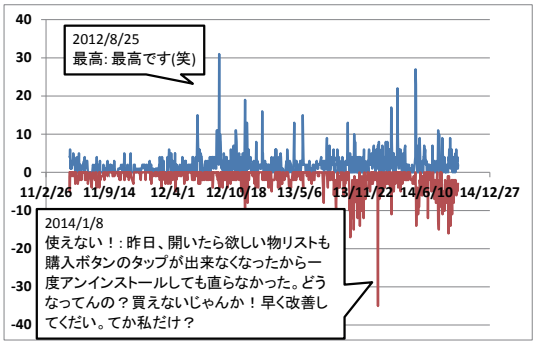
最後に，評価の変動表示機能について，Amazon 及び Cookpad のアプリを対象にした結果を図 4 に示す．縦軸の 0 以上の部分が SP を，縦軸の 0 以下の部分が NP の値の変動を表している．また実験では $\alpha = 1$ に設定し，それぞれ 1 つの最大ピークに対して，代表的なレビューを提示している．筆者らが各ピークにおけるレブー内容を確認したところ，Amazon アプリの「最高: 最高です（笑）」以外のレビューについては，確かに，ピーク時に発生している他の主なレビュー内容を忠実に表していることが確認できた．Amazon アプリに対する 2012 年 8 月 25 日のレビューについては，特に同内容を表しているようなレビューは無かったため，実験結果のような，あまり情報量を持たないレビューが抽出されたものと考える．

6. 考察と将来課題

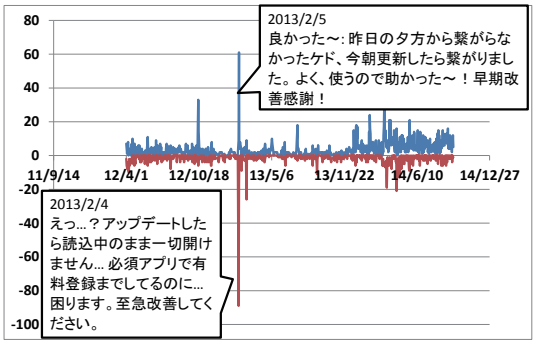
図 4(b) の例では，2013 年 2 月 4 日に不具合が発生しており，ユーザ評価が急激に下がっていることが分かる．しかし，2013 年 2 月 5 日に改善が見られると，早期対応に感謝するレビューが多数投稿され，高い評価が得られている．このように，不具合を検知した場合，一時的にユーザ評価が低下することは避けられないものの，早期対応によって評価を上げることができる．

将来課題として以下のことを考えている．

「スマートフォン」と「スマホ」等，同義語を適切に扱うことにより，より効果的に情報をユーザに提示できると考えられる．同義語の獲得に関する既存研究を利用することができる [15, 17]．



(a) Amazon



(b) Cookpad

図 4　ユーザ評価の変動提示
Fig. 4　Results of evaluation change

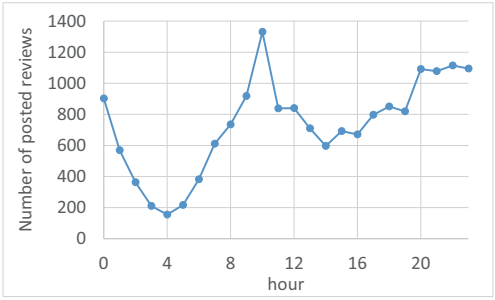


図 5　レビュー投稿時間の分布
Fig. 5　Distribution of time of review posting.

ユーザ評価の変動提示では，1 日の始まりを 0 時 0 分 0 秒，終わりを 23 時 59 分 59 秒としている．しかし，ユーザのレビュー投稿行動が最も落ち込むこむ時間帯を境にして 1 日を区切るほうが，より適切に表示ができるものと考えられる．たとえば，実験に利用したカレンダーアプリ群のレビュー投稿時間は，午前 4 時頃が最も少ない（図 5）．

レビューには，ユーザが利用している機種名が記載されていることがある．これは，各ユーザが，機種依存の問題である可能性を考慮して，自発的に記載しているものである．機種名が書かれたレビューを，アプリ横断的に収集し，機種依

存の問題を抽出することができると考えている．たとえば，Android OS における機種依存問題を吸収することを目指すプロジェクトである Android-Device-Compatibility [10] によると，特定の機種で入力文字数が勝手に設定されたり，写真の保存箇所が機種ごとに異なっていたり，特定の機種のときだけクラッシュするライブラリがあったりすることが分かっている．このような機種依存問題は，OS がバージョンアップされるたび，また，新しい機種が発売されるたびに起こり得る問題であるため，レビューを分析することで自動的に収集することができれば，有用性は高いと考えられる．

7. おわりに

本論文では，スマートフォンアプリケーション開発者支援システムを提案した．スマートフォンアプリケーション提供サイトに投稿されているレビューを解析することにより，機能/非機能の要求抽出，不具合検知，評価の変動表示を行う．Amazon アプリ，Cookpad アプリ，カレンダーアプリに対して行った実験により，ある程度，開発者にとって有用な情報を抽出できたと考える．今後の課題として，まず，開発者に実際に使ってもらって評価を行う必要がある．また，前章で述べたような，機能拡張を行う．

謝辞 本研究は JSPS 科研費 24300005, 26330081, 26870201 の助成を受けたものです．

参考文献

[1] AppTornado GmbH: Number of available android applications, `http://www.appbrain.com/stats/number-of-android-apps`.
[2] Bano, M. and Zowghi, D.: A systematic review on the relationship between user involvement and system success, *Information and Software Technology*, p. In press (2014).
[3] Blei, D. M., Ng, A. Y. and Jordan, M. I.: Latent dirichlet allocation, *The Journal of Machine Learning Research*, Vol. 3, pp. 993–1022 (2003).
[4] Chen, N., Lin, J., Hoi, S. C. H., Xiao, X. and Zhang, B.: AR-miner: mining informative reviews for developers from mobile app marketplace, *Proc. ICSE*, pp. 767–778 (2014).
[5] Cinque, M.: Enabling on-line dependability assessment of Android smart phones, *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, pp. 286–291 (2011).
[6] Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J. and Sadeh, N.: Why people hate your app: making sense of user feedback in a mobile app store, *Proc. ACM KDD*, pp. 1276–1284 (2013).
[7] Galvis Carreño, L. V. and Winbladh, K.: Analysis of user comments: an approach for software requirements evolution, *Proc. ICSE*, pp. 582–591 (2013).
[8] Griffiths, T. L. and Steyvers, M.: Finding scientific topics., *Proc. National Academy of Sciences of the United States of America*, Vol. 101, No. Suppl 1, pp. 5228–5235 (2004).
[9] Jones, S., Poulsen, A., Maiden, N. and Zachos, K.: User roles in asynchronous distributed collaborative idea generation, *Proc. ACM Conference on Creativity and cognition (C&C)*, pp. 349–350 (2011).
[10] mixi Inc: GitHub リポジトリ, `https://github.com/mixi-inc/Android-Device-Compatibility`.
[11] Seyff, N., Graf, F. and Maiden, N.: Using Mobile RE Tools to Give End-Users Their Own Voice, *Proc. IEEE International Requirements Engineering Conference (RE)*, pp. 37–46 (2010).
[12] Seyff, N., Maiden, N., Karlsen, K., Lockerbie, J., Grünbacher, P., Graf, F. and Ncube, C.: Exploring how to use scenarios to discover requirements, *Requirements Engineering*, Vol. 14, No. 2, pp. 91–111 (2009).
[13] Sutcliffe, A. and Sawyer, P.: Requirements elicitation: Towards the unknown unknowns, *Proc. IEEE International Requirements Engineering Conference (RE)*, pp. 92–104 (2013).
[14] 鵜飼孝典， 林晋平，佐伯元司：要求獲得におけるステークホルダの偏りと不足を検出する可視化ツール，情報処理学会論文誌，Vol. 53, No. 4, pp. 1448–1460 (2012).
[15] 吉田稔，中川裕志， 寺田昭：コーパス検索支援のための動的同義語候補抽出，人工知能学会論文誌，Vol. 25, No. 1, pp. 122–132 (2010).
[16] 荒井大輔， 堀賢治，吉原貴仁：Android アプリ可用率の調査と安定化手法の提案，電子情報通信学会技術研究報告，Vol. 111, No. 469, pp. 103–108 (2012).
[17] 内海慶， 小町守：ウェブ検索クエリログとクリックスルーログを用いた同義語獲得，情報処理学会論文誌. データベース， Vol. 6, No. 1, pp. 16–28 (2013).