

機能に基づいたアプリ検索手法の提案

上西達也

背景、目的

新規アプリ普及の課題

- スマートフォンなどのアプリケーション(アプリ)は大量に出回っており、新規アプリも日々開発、リリースされている
- ユーザはそこから自由に選んで端末にインストールするが、**すでに人気のアプリを採用するユーザ**もいれば、**自ら好みのアプリを探して採用するユーザ**もいる
- 新規アプリを普及させたいときは、まず **自らアプリを探すユーザ**にアプリを見つけてもらう必要がある

自らアプリを探すユーザ

Everett M. Rogersは新しいアイデアや技術の採用が早い順にユーザを5つに分類した(イノベーション普及学)

1. **イノベーター**：冒険心にあふれ、新しいものを進んで採用する(2.5%)
2. **アーリーアダプター**：流行に敏感で、自ら情報収集する(13.5%)
3. **アーリーマジョリティ**：新しいものを取り入れるが比較的慎重(34.0%)
4. **レイトマジョリティ**：周囲の大多数を見てから同じ選択をする(34.0%)
5. **ラガード**：流行や世の中の動きに関心が薄く、最も保守的(16.0%)

アーリーアダプターの特徴

- 自らの利益を考慮しながら採用する
- アプリ購入前にその機能を把握し、求めている機能を持ったアプリを選びたい欲求があると考えられる

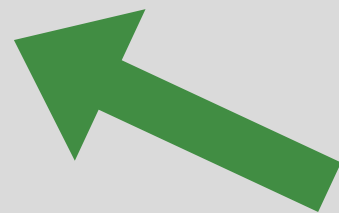
【最新】シンプルエディタ



【人気】高機能エディタ



無駄な機能に興味がない
アーリーアダプター



現在のアプリ検索システムの問題点

1. すでに人気のあるアプリを上位に表示する傾向がある※1
 - ダウンロード数が少ない**新規のアプリ**などはたとえ優れていてもアーリーアダプターの目に触れにくい
2. タイトルや説明文などの質によって順位が大きく変わる
 - アプリストア最適化 (ASO) が順位操作に効果を上げている現状がある※1
 - 全く機能の違う偽アプリが上位になることも※2

※1 : App Annie (www.appannie.com)

※2 : 「偽アプリに気を付けて: ポケモンGO便乗不正アプリが3900万件ダウンロード、Google Playでの配布も - ITmedia PC USER」(www.itmedia.co.jp/pcuser/articles/1608/18/news091.html)

アプリの機能に基づく検索の必要性

- アーリーアダプターは人気よりも求めている機能があるか（利益があるか）に着目する
- 純粋に機能に基づいたアプリ検索システムがあれば、アーリーアダプターが求めている機能を持ったアプリを見つけやすくなる
- 新規でもよりよいアプリが普及し、ソフトウェア業界の活性化に繋がる

アプローチ

対象とするユースケース

既知のアプリを入力として機能面で類似するアプリを検索する場合

- 欲しい機能があるもの
- いない機能がないもの

解析対象の制約と解決策(1)

制約

アプリの多くはソースコードが公開されていない



解決策

公開されている情報(**実行可能ファイルの情報**など)だけを使う

解析対象の制約と解決策(2)

制約

実行可能ファイルは難読化されていることが多い



解決策

多くの難読化ツールが標準API(メソッド)呼び出しまでは難読化しないことを利用する

解析アプローチ

一般的なアプリ開発では、標準APIを組み合わせて目的の機能を実装する



各クラス内の標準API呼び出しの組み合わせの類似性(共起関係の類似性)は機能と関連があると考えられる



潜在的意味インデキシング(LSI)を用いた次元圧縮によるインデックス化を行って検索する

潜在的意味インデキシング (LSI)

- 文書群とそれらに含まれる単語群から、関連した概念の集合を生成して、その関係进行分析する技術
- ベクトル空間モデルを利用した自然言語処理の技法の1つ
- 各文書における単語の出現を表した**文書-単語行列** (Bag of Wordsモデル) から、**文書-概念行列**、**単語-概念行列**を生成する

文書-単語行列

	d1	d2	d3	d4
会場	1	1	1	0
店	0	0	0	1
自動車	0	1	0	0
自転車	0	0	1	1
行く	1	1	1	1
車	1	0	0	0

このままでは言い換え表現や表記ゆれなどに対応できない

単語の持つ潜在的意味を軸にした行列を作成して対応する

特異値分解

文書-単語行列(X)を、直交行列U、Vと対角行列Σに分解する

$$X = U\Sigma V^T$$

ここで、直交行列U、Vがそれぞれ文書、単語の潜在的意味を軸にした行列(基底行列)になっている

特異値分解の例

d1と**d3**は共通の潜在的意味を持つ



$$\begin{array}{l} \text{医者} \\ \text{病院} \end{array} \begin{array}{cc} \mathbf{d1} & \mathbf{d3} \end{array} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix} = \begin{bmatrix} -0.707 & 0 \\ 0 & 1 \\ -0.707 & 0 \end{bmatrix} \begin{bmatrix} 3.16 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -0.447 & 0 \\ 0 & 1 \\ -0.894 & 0 \end{bmatrix}^T$$



医者と**病院**は共通の潜在的意味を持つ

LSIでは、これら文書-概念行列、単語-概念行列を、検索用のインデックス(索引)とする

解析アプローチ(再掲)

一般的なアプリ開発では、標準APIを組み合わせて目的の機能を実装する



各クラス内の標準API呼び出しの組み合わせの類似性(共起関係の類似性)は機能と関連があると考えられる



潜在的意味インデキシング(LSI)を用いた次元圧縮によるインデックス化を行って検索する

実装、ケーススタディ

検索対象

- **Androidアプリ**を対象とする
 - 実行可能ファイルはAPKファイル(JARファイルに変換できる)
- JARファイルから各クラス内の標準API呼び出しの組を抽出するために、ソフトウェアバースマーク取得ツール**Stigmata**を応用する

Stigmata

ソフトウェアの盗用を検出するために開発されたツール

ソフトウェアバースマークと呼ばれるソフトウェア固有の特徴を検出する

ソフトウェアバースマークの例

- 使用クラス
- メソッド呼び出し列 (← これを応用する)
- コードクローン

検索手順

1. 各APKファイルから、クラスごとに標準API呼び出しの組を抽出
2. クラスを文書、標準API呼び出しを単語としてBag of Wordsを作成
3. LSIを適用
4. 既知のアプリ(検索の入力)の各クラスと同じ機能を持つと推定されるクラスをより多く持つアプリを提示

1. 標準API呼び出しの抽出



クラスごとに抽出する

2. Bag of Wordsを作成

Class A

A.m1()
A.m2()
B.m1()

Class B

A.m1()
B.m1()
C.m2()

Class C

A.m1()
C.m1()
C.m2()

BoW化

	A.m1()	A.m2()	B.m1()	C.m1()	C.m2()
ClassA	1	1	1	0	0
ClassB	1	0	1	0	1
ClassC	1	0	0	1	1

3. LSIを適用

	A.m1()	A.m2()	B.m1()	C.m1()	C.m2()
ClassA	1	1	1	0	0
ClassB	1	0	1	0	1
ClassC	1	0	0	1	1



グループ1

ClassA
ClassB
ClassC

グループ2

ClassA
ClassB

グループ3

ClassB
ClassC

検索手順(再掲)

1. 各APKファイルから、クラスごとに標準API呼び出しの組を抽出
2. クラスを文書、標準API呼び出しを単語としてBag of Wordsを作成
3. LSIを適用
4. 既知のアプリ(検索の入力)の各クラスと同じ機能を持つと推定されるクラスをより多く持つアプリを提示

ケーススタディ

ここにきて出力が全て0.0に...

もう少しかかります...