

標準API呼び出しの共起関係を利用した アプリ検索手法の提案

オープンソースソフトウェア工学研究室

60181007 上西達也

背景

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

アプリの選び方はユーザによって様々

- すでに人気のアプリを採用するユーザ
- 自ら好みのアプリを探して採用するユーザ

新規アプリ普及の課題

開発者が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ーの特徴

- 流行に敏感で、新しければとりあえず採用してみる姿勢、冒険心がある^[1]
 - 利益を考慮しながら採用する(リスク許容度が低い)人も多い^[1]
- ↓
- 類似アプリ群の中から最も自分に合ったアプリを選びたい
 - ただし、どんな機能が欲しいか具体的に記述できない(アプリを見て初めて自分の求めている機能を認識する)

現在のアプリ検索システムの問題点

- ユーザは自身の求める機能を適切にクエリ化できず、検索できないことがある
- アプリの説明文などにユーザの求める機能と合致する説明が書かれているとは限らない
- アプリ購入後に、機能に不満を感じることも



アプリ購入前に機能が類似したアプリ同士を比較できるとよい

目的

- 機能的特徴が類似するアプリの検索手法を提案すること
- アプリの機能的特徴の差分取得手法を提案すること

※機能的特徴…本研究ではクラスやメソッドの利用状況と定義

本研究の前提

- 既存のソフトウェア比較手法はソースコードが手に入ることを前提としている
- アプリの多くはソースコードが公開されていない
- 実行可能ファイルから抽出できる情報などを使った手法が必要



本研究では、解析ツールStigmataを応用する

Stigmata

- ソフトウェアの盗用を発見、証明するための研究で開発されたJava向け解析ツール
- クラスファイルやJARファイルから、標準API呼び出し列などを抽出できる

(この辺に標準API呼び出し列の例を書く)

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呼び出しの共起関係の類似性は機能と関連があると考えられる



標準API呼び出しの共起関係を利用したアプリ検索手法を構築

提案手法

Androidアプリを対象とする

(流れ図で書く)

1. アプリを解析
2. 共起関係コーパスを作成
3. アプリ間類似スコア、差分の取得

1. アプリ解析

1. アプリのAPKファイルをJARファイルに変換
2. Stigmataを用いて標準API呼び出し列の集合を抽出
3. 出力はアプリごとにCSV形式で得られる

(ここに出力結果の表を例示)

2. コーパス作成(1/2)

クラスと標準API呼び出しの共起行列 (Bag of Wordsモデル)

(ここに共起行列作成の流れを例示)

2. コーパス作成(2/2)

1. 標準API呼び出しをフィルタ

- 単一のアプリのみに存在する標準API呼び出し
- 過半数のアプリに存在する標準API呼び出し

2. クラスをフィルタ

- アプリに含まれる標準クラス(android.*)
- 標準API呼び出しが1つもないクラス

3. tf-idfを適用

4. LSI(潜在的意味インデキシング)を適用

3-1. アプリ間類似スコアの取得

本研究における定義

- 類似する … 閾値以上の類似度を持つ
 1. 2アプリに含まれる各クラス間の類似度を算出
 2. 各クラスにとって最大の類似度のみを選択
 3. 閾値以上の類似度を選択
 4. 上に当てはまる類似度の総和を類似スコアとする
- (図で説明?アニメーションがわかりやすい?)

3-2. 差分の取得

本研究における定義

- 類似する … 閾値以上の類似度を持つ
 - 差分 … 類似したクラスが1つもないクラス
1. 2アプリに含まれる各クラス間の類似度を算出
 2. 閾値以上の類似度を持つクラスを除外
 3. 類似したクラスが1つもないクラスを差分とする
- (図で説明?アニメーションがわかりやすい?)

提案手法(再掲)

(初出より少し詳しくめの説明を付けた流れ図に)

1. アプリを解析
2. 共起関係コーパスを作成
3. アプリ間類似スコア、差分の取得

ケーススタディ

目的

提案手法の性能(機能が類似したアプリ同士をどの程度比較できるか)を確かめる

Research Question

1. LSI適用による処理速度への影響はどの程度か
2. アプリ間類似スコアと差分の出力は適切か
3. アプリ間類似スコアとクラス間類似度の関係調査(相関?)

実験対象

対象：Googleが提供するAndroidサンプルアプリ

(ここに表?)

RQ1. LSI適用の影響はどの程度か

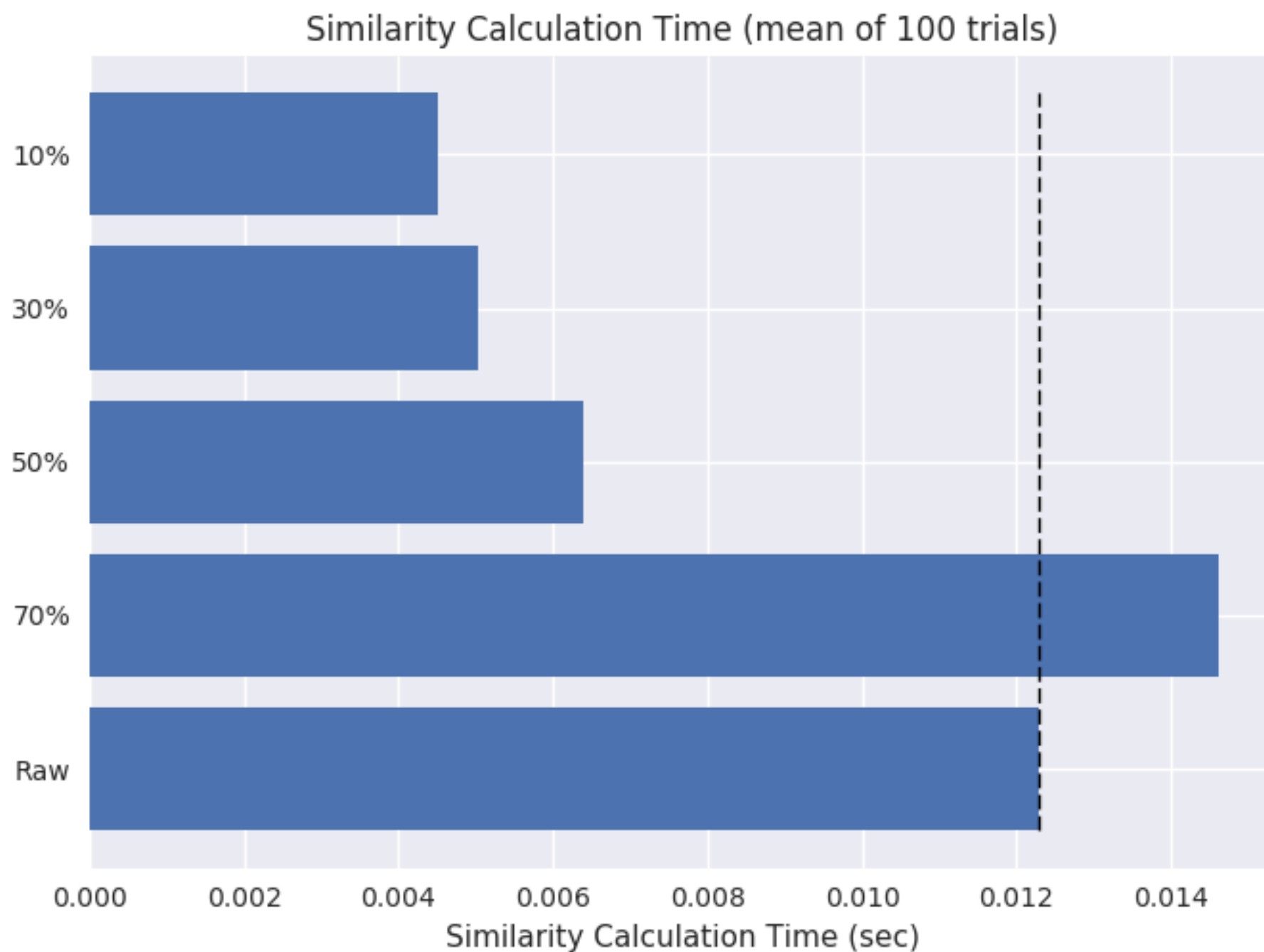
目的

LSI適用の目的であった速度向上は実現されているか

評価方法

クラス間類似度計算部分の処理時間を計測

RQ1. LSI適用の影響はどの程度か



RQ1. LSI適用の影響はどの程度か

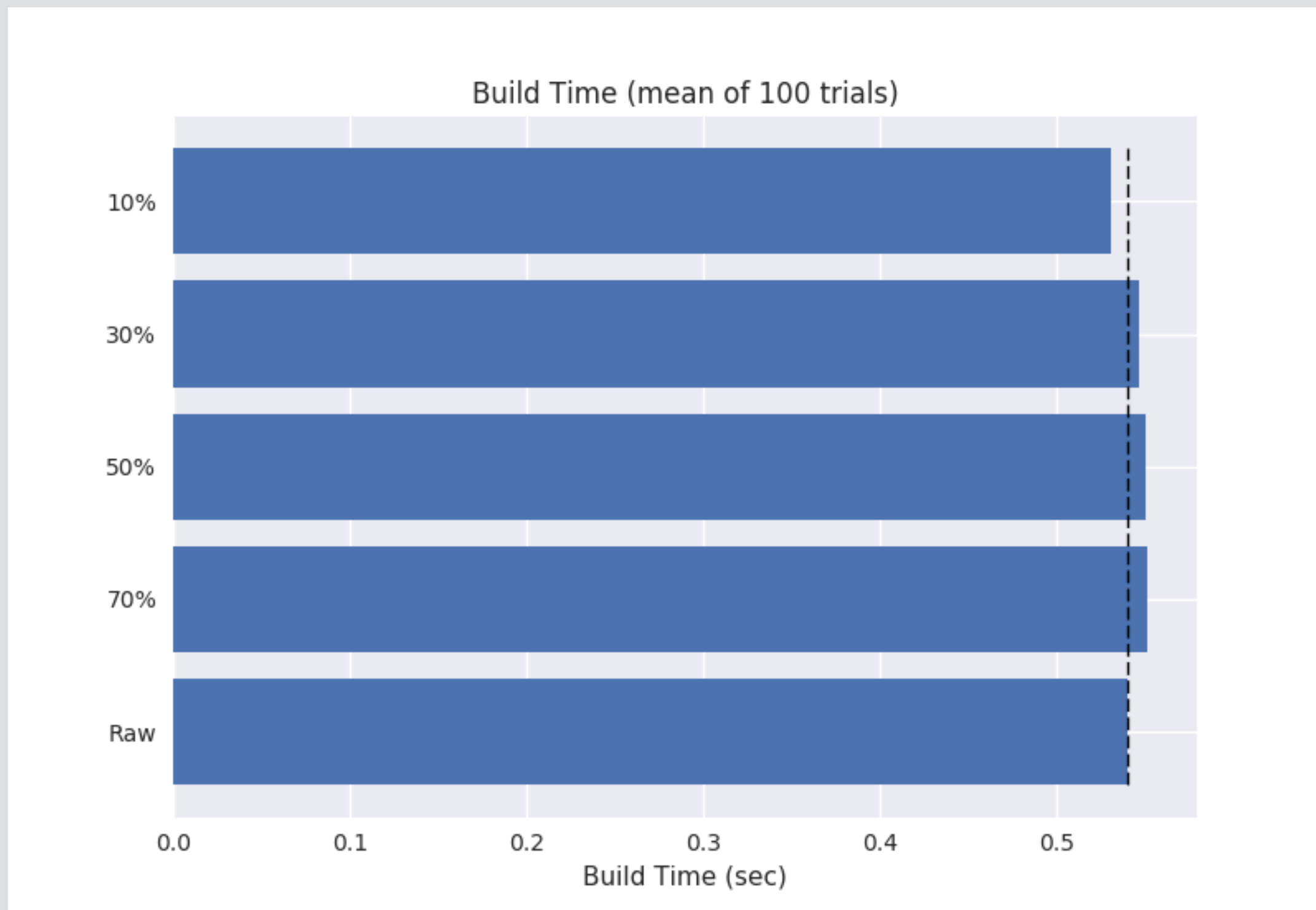
目的(追加実験)

LSI適用によるコーパス作成時間への影響はどの程度か

評価方法(追加実験)

共起関係コーパス作成部分の処理時間を計測

RQ1. LSI適用の影響はどの程度か



RQ1. LSI適用の影響はどの程度か



RQ1. LSI適用の影響はどの程度か

結果

- LSIによる行列の圧縮率が大いほど(約50%以下)、クラス間類似度の計算時間は半分ほどに短縮された
- LSIによる行列の圧縮率が大いと(約70%以上)、かえって処理時間が延びた
- コーパス作成時間はLSI適用の有無によって大きな変化は見られなかった(むしろ早くなった例も)

RQ2. アプリ間類似スコア、差分は適切か

目的

評価方法

結果

RQ3. アプリ間類似スコアとクラス間類似度の関係

目的

評価方法

結果

考察

まとめ