Androidアプリの説明文とプライバシー情報アクセスの相関分析

渡邉 卓弥 †　秋山 満昭 ‡　酒井 哲也 †　鷲崎弘宜 †　森 達哉 †

† 早稲田大学 基幹理工学部　‡NTT セキュアプラットフォーム研究所
169-8555　東京都新宿区大久保 3-4-1　180-8585　東京都武蔵野市緑町 3-9-11

あらまし スマートフォンユーザーの意図しないプライバシー情報の漏洩を防ぐために，開発者はアプリが端末からどのような情報を取得するか十分に開示すべきである．本研究は，プライバシー情報の取得をユーザーに通知する手段としてアプリの説明文に着目する．マーケットにおける説明文の現状を調査するため，Android アプリのコード解析と機械学習による文書分類を組み合わせたフレームワークを開発した．20 万個の検体にフレームワークを適用した結果，プライバシー情報へのアクセスについて説明文で言及している割合は全体の 4%-35%であることが判明した．さらに抽出された検体を詳細に分析し，アプリが暗黙のうちにプライバシー情報にアクセスを試みる主な要因を解明した．

Analyzing the inconsistency between words and actions of Android apps

Takuya Watanabe†　Mitsuaki Akiyama‡　Tetsuya Sakai†　Hironobu Washizaki†
Tatsuya Mori†

†Waseda University　‡NTT Secure Platform Laboratories
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555　3-9-11 Midoricho, Musashino-city, Tokyo 180-8585

Abstract In order to prevent unintended leakage of private information for smartphone users, developers should fully disclose what kind of information their applications will obtain from the device. In this study, we focus our attention on the description of a given application as a means to inform the user of the use of private information. To investigate the current status of application descriptions in the market, we have developed a framework which combines code analysis and machine-learning-based document classification for Android applications. As a result of applying our framework to a sample of 200,000 applications, we found that only 4-35% of the descriptions mention the application's access to private information. We closely examined the detected applications and identified the primary reasons why they attempt to access private information without notifying the user.

1　はじめに

スマートフォンの世界的な普及にともない，モバイルアプリが我々に及ぼす影響は計り知れないものとなった．Gartner 社 [1] の報告によると，モバイルアプリのダウンロード数は 2014 年のうちに 1,380 億回にも及ぶと試算されている．スマートフォンがもたらす利便性が世界中の人々の生活を豊かにする一方で，モバイルアプリによる個人情報の漏洩が社会的な問題 [2, 3, 4] となっている．悪質なモバイルアプリは往々にして，ユーザーのあずかり知らぬところでプライバシー情報を収集する．

アプリマーケットの運営者は，モバイルアプリによるプライバシー情報の取得をユーザーに通知するための仕組みを設けている．インストール時に表示されるパーミッションの警告や，マーケットにリンクされるプライバシーポリシーの掲示である．これらの対策はプライバシー保護のため重要であるが，現状では十分に活用されているとはいえない．Felt ら [3] によれば，インストールの際にパーミッション警告を意識しているユーザーは 17%程度であり，また Chin ら [5] によれば，80%程度のユーザーがプライバシーポリシーを閲覧していなかった．

本研究はこのような背景を踏まえ，プライバシー情報の取得を通知する手段として「**アプリの説明文**」に焦点を当てる．アプリの説明文は公式，サードパー

ティ問わずあらゆるマーケットに設定されており，アプリを探す際にはキーワードによる検索の対象となる．説明文は未知のアプリの機能を，インストールする前に把握するための唯一の手段であるといえる．したがってユーザーは，パーミッションの警告やプライバシーポリシーよりも注意深く説明文を閲覧することが期待される．本研究の狙いは，以下に示す 2 点について解き明かすことである．

1. アプリマーケット上で，どの程度のアプリがプライバシー情報の取得について言及しているか．
2. アプリが説明文で言及することなくプライバシー情報の取得を試みる原因は何か．

本研究では ACODE (Analyzing COde and DEscription) と名付けるフレームワークを開発し，上記の問いの解明を目指す．ACODE は Android アプリを対象に，コード解析とテキスト分類を組み合わせたアプローチを行う．コード解析では，プライバシー情報を実際に取得するアプリを逆アセンブルコードの解析によって抽出する．テキスト分類では，説明文がプライバシー情報の取得について言及しているか否かを教師あり機械学習によって分類する．

解析の対象とするリソースとして連絡帳，位置情報，カメラの 3 種類を選択した．連絡帳と位置情報は最も狙われる [6] リソースであり，カメラは今後大きなプライバシーの脅威 [7] となり得るリソースである．説明文の言語は世界で広く使われる英語と中国語の 2 種類を対象とし，公式マーケットである Google Play とサードパーティマーケットから 10 万検体ずつ，計 20 万検体に ACODE を適用した．

本研究で得られた主な知見を以下に示す．

- 公式マーケットにて公開されているアプリのうち，連絡帳，位置情報，カメラへのアクセスについて説明文で言及している割合は 17-35%であった．
- サードパーティマーケットにて公開されているアプリのうち，連絡帳，カメラへのアクセスについて説明文で言及している割合は 17-20%であり，位置情報に至ってはわずか 4%であった．
- 言及なしにリソースへアクセスするアプリには，説明文が短い，クラス数が多い，一部の開発者が量産しているなどといった傾向が見られた．
- 特定の用途で機能を使う場合，多くの開発者はリソースへのアクセスについて言及しないことがわかった．言及されない用途としてはショッピングアプリに付随する QR コード認識機能などがある．
- 一部の自動生成サービスによって開発されたアプリは，本来不要であるはずのパーミッションや API を多数含んでいた．
- 言及なしに位置情報を取得するアプリのうち，55%-78%は広告ライブラリによるものであった．

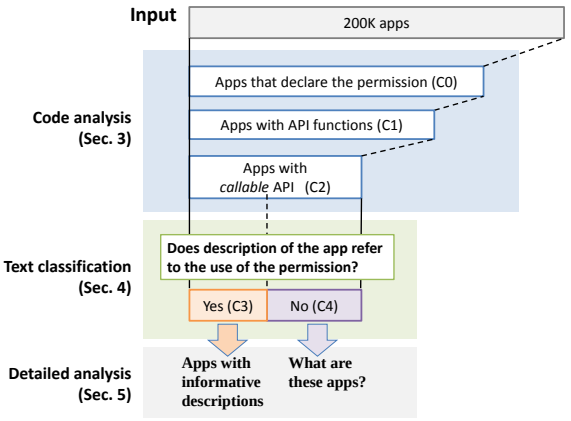

図 1: ACODE フレームワークの概要

本研究の最大の貢献は，膨大な数の検体を分析することで，モバイルアプリマーケット全般における説明文とプライバシー情報アクセスの相関性を明らかにしたことにある．上記の知見は，ユーザーのプライバシーに配慮したアプリマーケットを設計するために有用であり，またユーザーおよび開発者のプライバシー意識を向上させるための足がかりとなる．

本論文の構成は以下の通りである．はじめに 2 章では ACODE フレームワークの概要を述べる．3 章ではコード解析の具体的な方法を述べ，4 章ではテキスト分類の具体的な方法を述べる．5 章では ACODE をマーケット上のアプリに適用した結果と考察を述べ，6 章では関連研究との比較を行う．最後に 7 章で本論文のまとめと今後の展望を述べる．

2 ACODE フレームワーク

本章では，ACODE フレームワークの概要を示す．1 章で述べた通り，ACODE はコード解析とテキスト分類を組み合わせたアプリ抽出を行う．図 1 に ACODE によるアプリ抽出フローを示す．コード解析では，まずプライバシー情報に関わる特定のパーミッションを要求しているアプリを抽出する ($C0$)．その中から，パーミッションに紐付けられた API をコードに含むアプリを抽出する ($C1$)．最後に，関数のコールグラフを追跡することで実際に API が使用されているアプリを抽出する ($C2$)．テキスト分類では，説明文にプライバシー情報へのアクセスを通知する文言が含まれているもの ($C3$) と，含まれていないもの ($C4$) を自動的に分類する．ACODE の目標は，以上の抽出フローを多数のアプリに適用し，プライバシー情報の取得をユーザーに通知する手段として説明文の有用性を定量化することである．$C3$ に分類されるアプリを数えることで，現状でどの程度のアプリがプライバシー情報の取得について言及しているかを計測する．また $C4$ に分類されるアプリ

の特徴を分析することで，アプリが暗黙のうちにプライバシー情報を取得する原因を調査する．

3 コード解析

本章では，コード解析によってプライバシー情報へのアクセスを行うアプリを抽出する方法を示す．パーミッションフィルタリング，API フィルタリング，コールグラフ解析の 3 つの段階を踏むことで，必要な計算量を軽減しつつ実際にプライバシー情報へアクセスを行うアプリを抽出する．

3.1 パーミッションフィルタリング

特定のパーミッションを要求するアプリを抽出する方法を示す．Android アプリでは，AndroidManifest.xml というマニフェストファイルの中でパーミッションが宣言される．aapt [8] によってマニフェストを分析し，パーミッションの有無を確認する．本研究で分析の対象としたリソースは，詳細位置情報，連絡帳，カメラの 3 種類で，対応するパーミッションは *ACCESS_FINE_LOCATION*，*READ_CONTACTS*，*CAMERA* である．各パーミッションの詳細は，公式マニュアル [9] に記載されている．

3.2 API フィルタリング

開発者が不要なパーミッションを宣言してしまうケースが少なくない [10] ため，リソースの使用を調べるにはパーミッションの確認だけでは不十分である．そこで本節では，リソースを使用する API を含むアプリを抽出する方法を示す．Android ではパーミッションごとに API が対応付けられており，これら API の有無を確認することでプライバシー情報へのアクセスを判別する．ここで API とは，位置情報を取得するための関数や連絡帳の場所を示す URI などを指す．連絡帳のような端末内の静的データは，コンテントプロバイダと呼ばれるデータベースによって管理される．パーミッションと API の対応表として，Au らによって開発された PScout [11] を用いた．apktool [12] によってアプリを smali [13] 形式に逆アセンブルした後，文字列検索によってコード内に各種 API が含まれるか確認する．

3.3 コールグラフ解析

Android アプリでは，外部ライブラリのインポートによって使用しないコードを含むケースが見受けられる．前節のフィルタリングは，こういった到達不能な API を含むアプリを抽出してしまう．本節では，関数コールグラフを追跡し，実際にプライバシー情報へアクセスを行うアプリを抽出する方法を示す．

図 2 にコールグラフの追跡アルゴリズムを擬似コードで記述した．このアルゴリズムはパーミッションに紐付く API を起点に，メソッドの呼び出し元を順次辿る．メソッドがコンストラクタ内で呼び出されていた場合には，該当クラスのインスタンス生成を辿る．追跡の結果，4 行目の ORIGIN に含まれるクラスのサブクラスに到達する場合，アプリは API に到達可能であるとみなす．到達しない場合，アプリは API を利用していないため解析対象から除外する．ORIGIN は，Application，App Components，Layout の 3 つのクラスから構成される．Application はアプリ起動時に必ず呼ばれるクラスである．App Components は Android アプリの動作を決める重要なクラスであり，Activities，Services, Content providers, Broadcast receivers の 5 つから構成される．Application および App Components に該当するクラスはマニフェストファイルに記述される．Layout は XML を用いてデザインとコードを分離するための仕組みであり，主に広告ライブラリに実装されている．5 行目の getAU は PScout をもとに与えられたパーミッションに紐付いた API 群を返す．21 行目の refFunctions はメソッド f の呼び出し元のメソッドを返す．呼び出し元のメソッドを走査するために，androguard [14] を改良したものを使用した．Android SDK を継承したクラスでオーバーライドされたメソッドはアプリのコード内に記述されていなくても呼び出すことができるため，コールグラフ上で呼び出し関係にないことに注意する．その場合はメソッド呼び出しの代わりにインスタンス生成を追跡するというヒューリスティックを用いた．ヒューリスティックにより，並列処理やイベントハンドラのようにメソッド呼び出しを記述しない処理を追跡することができる．このアルゴリズムを 16 行目から 19 行目に示す．

4 テキスト分類

本章では，ACODE で使用するテキスト分類の方法を示す．テキスト分類の目的は，プライバシー情報へのアクセスに言及している説明文とそうでない説明文の 2 つに分類することである．Bag-of-Words (BoW) モデル，tf-idf，教師あり機械学習を組み合わせ，自然言語処理の標準的な二値分類器を構築する．BoW はテキストの文法や順序を無視し単語の集合体として扱うモデルである．tf-idf は単語の出現頻度に基づく重み付けである．教師あり機械学習のアルゴリズムには Support Vector Machine (SVM) を用いた．さらに分類精度を高めるため，自己相互情報量を用いた単語ヒューリスティックを適用した．この手法を 3 種類のリソースと 2 種類の言語に対して適用し，正答率 92%以上，偽陽性率 5.6%以下という

```
1: INPUT
2: p : a permission
3: a : an application (APK)
4: ORIGIN =  [Application, App Components, Layout]
5: list = getAU(p)   # list of APIs associated with p
6: done = []          # empty list
7:
8: WHILE list is not empty DO
9:     f = list.pop()
10:    IF f is in done:
11:        skip the function
12:    ENDIF
13:    IF f.parentClass is in ORIGIN:
14:        RETURN True
15:    ENDIF
16:    IF (f.parentClass inherits Android SDK)
17:        AND (f is not init)
18:        AND (f is not a static method):
19:        list.append(f.parentClass.init)
20:    ELSE IF (f is referred):
21:        list.append(f.refFunctions)
22:    ENDIF
23:    done.append(f)
24: ENDWHILE
25: RETURN False
```

図 2: 関数コールグラフを用いた API 呼び出し追跡のアルゴリズム

高い精度で分類することに成功した．4.1 節では機械学習に用いたラベルの生成および選択方法を示し，4.2 節では自然言語処理のためにテキストに施した前処理を示す．4.3 節では SVM の説明と，クロスバリデーションによる性能評価の方法と結果を示す．4.3 節では，自己相互情報量を使った単語ヒューリスティックについて説明する．

4.1 ラベルの生成と選択

説明文のテキストを分類するためのラベルを生成した手順を示す．SVM の識別関数を適用するための訓練データとして，言及ありと言及なしの 2 種類のラベルを持った説明文を用意する．中国語と英語を母国語とする留学生 12 名の協力のもと，手動でマーケット上のアプリの説明文を分類した．参加者のうち 6 名は英語，もう 6 名は中国語の説明文に対してラベル付けを行った．

表 1 にラベル付けのため用意した説明文の内訳を示す．マーケットから無作為に抽出した場合，言及ありの説明文と比べて言及なしの説明文が圧倒的に多く不均衡によって機械学習の精度が著しく低下してしまうことがわかった．そこで 3.1 節のフィルタリングを利用して，分類対象のパーミッションが宣言されているアプリのみラベル付けの対象とした．各言語と各パーミッションに 1,000 個ずつ説明文を用意し，協力者は 3 つのパーミッションについて 500 個ずつの説明文を分類した．すなわち表の通り，計 6,000 個の説明文に対して 3 名ずつの協力者が担当し，18,000 個のラベルを収集した．

表 1: ラベル付けの対象とした説明文の内訳

	English			Chinese		
	Location	Contact	Camera	Location	Contact	Camera
# of descriptions	1,000	1,000	1,000	1,000	1,000	1,000
# of labels	3,000	3,000	3,000	3,000	3,000	3,000

表 2: 最終的に訓練データとした説明文の内訳

	English			Chinese		
	Location	Contact	Camera	Location	Contact	Camera
# of total descriptions	783	718	712	923	696	856
# of positive descriptions	76	125	165	25	62	130
# of negative descriptions	707	593	547	898	634	726

最終的に 3 名の協力者による評価が一致しないものを取り除き，すべて一致したもののみを訓練用ラベルとした．これは協力者の認識の違いによる分類の誤差を避け，より正確な分類を行うためである．次節以降で SVM の訓練用ラベルとして使用する説明文の内訳を表 2 に示す．すべてのカテゴリーで少なくとも 700 程度のラベルを学習に使った．

4.2 テキストの前処理

SVM を適用するため説明文に施した前処理を示す．まず説明文に対して単語分割，ステミング，ストップワードの除去を行い，その後 tf-idf による重み付けを用いて説明文の特徴ベクトルを生成した．

単語分割は，テキストを単語単位で分割し単語の集合に変換する．英文テキストでは単語間の空白やコンマ，ピリオドを区切りとして分割すればよい．中文テキストでは区切り文字が存在しないので，辞書データをもとに単語を分割する必要がある．本研究では単語分割のために KyTea [15] を利用した．ステミングは，活用によって語形が変化した単語を 1 つにまとめる処理である．英語のステミングのために nltk [16] を利用した．中国語には活用形がないため，ステミングは不要である．ストップワードは，前置詞や限定詞のようにあらゆる文書で使用され，分類の役に立たない単語を意味する．ストップワードのリストとして，英語では nltk [16]，中国語では Imdict [17] を用いた．ステミングやストップワードの除去によって，分類精度の向上が見込まれる．

以上の処理をテキストに施した後，各単語の tf-idf を成分とし説明文ごとの特徴ベクトルを生成した．ここで tf-idf は文書内の出現頻度に応じて単語を重み付けするための値であり，情報検索でよく利用される．tf-idf の詳細は引用 [18] に詳しい．

4.3 機械学習によるテキスト分類

表 2 に示したラベル付き説明文を訓練データとして，未分類の説明文を二値分類する．1 を言及あり，−1 を言及なしとした i 番目のラベル $y_i \in \{1, -1\}$ と

表 3: 交差検証によって測定した SVM の分類精度

		ACC		FPR		FNR	
		mean	std	mean	std	mean	std
English	Location	0.963	0.017	0.015	0.013	0.241	0.154
	Contact	0.915	0.033	0.056	0.027	0.218	0.122
	Camera	0.921	0.031	0.025	0.020	0.255	0.106
Chinese	Location	0.981	0.013	0.007	0.011	0.413	0.341
	Contact	0.959	0.022	0.022	0.019	0.220	0.177
	Camera	0.951	0.025	0.018	0.015	0.220	0.117

その特徴ベクトル $\mathbf{x}_i$ を用いて教師あり機械学習を行う．ここでは高次元な特徴に対する学習の高速化を図るために，L1 正則化付き，L2-損失関数の線形 SVM を採用した．識別関数 $\widehat{y_i} = f(\mathbf{x}_i)$ を線形関数によって $f(\mathbf{x}_i) = \mathrm{sgn}(\boldsymbol{\theta}^T \mathbf{x}_i)$ のように構成する．$\mathrm{sgn}(x)$ は符号関数，$\boldsymbol{\theta}$ は識別関数のパラメタである．L1-拘束付き，L2-損失関数の SVM のパラメタ学習は以下の最適化問題を解くことで得られる．

$$\widehat{\boldsymbol{\theta}} = \mathrm{argmin}_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{\theta}\|_1 + C \sum_{i=1}^{N} \left(\max(0, 1 - y_i \boldsymbol{\theta}^T \mathbf{x_i}) \right)^2 \right\}$$

$\|\cdot\|_1$ は L1 ノルムであり，$C > 0$ はソフトマージンのペナルティを制御するパラメタである．C は交差検証を用いた探索によって最適な値に決定した．

つづいて SVM による分類の精度を評価する．線形 SVM の実装として LIBLINEAR [19] を使用した．精度の評価には，ラベル付きデータを訓練データとテストデータに分割する交差検証を行う．乱数の偏りを避けるため，シードを 10 回変えて 10 分割の交差検証を行い，計 100 回の試行から平均値を計測する．表 3 に精度測定の結果を示す．ここでは正答率 (ACC)，偽陽性率 (FPR)，偽陰性率 (FNR) の 3 つの尺度で計測した．FPR は言及なしの説明文を言及ありと判定するエラー率であり，FNR は言及ありの説明文を言及なしと判定するエラー率である．すべてのカテゴリにおいて，ACC は 0.9 以上，FPR は 0.06 未満と良好な値となった．一方 FNR は FPR と比べてやや高い値となった．

4.4 単語ヒューリスティック

アプリが言及せずプライバシー情報の取得を試みる原因を解明するためには FPR を低く抑えることが肝要である．一方で言及なしの説明文は言及ありの説明文よりも極めて多いため，FNR の高さは解析に大きな影響を与えないが，低いに越したことはない．FPR を低く維持しつつ FNR を下げるため，特定の単語を含む説明文を強制的に言及ありと分類するヒューリスティックを適用した．

単語の選択方法は次の通りである．まず言及ありとラベル付けされた説明文に 3 回以上出現する単語を抽出する．次に言及ありのクラスに対する単語

表 4: SVM と単語ヒューリスティックを組み合わせた際の分類精度

		ACC		FPR		FNR	
		mean	std	mean	std	mean	std
English	Location	0.961	0.020	0.020	0.017	0.211	0.146
	Contact	0.915	0.033	0.056	0.027	0.218	0.122
	Camera	0.924	0.031	0.025	0.020	0.245	0.109
Chinese	Location	0.984	0.013	0.007	0.011	0.333	0.348
	Contact	0.960	0.023	0.027	0.021	0.176	0.156
	Camera	0.951	0.025	0.018	0.015	0.217	0.115

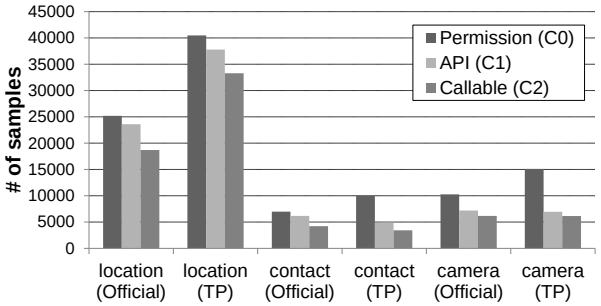

図 3: コード解析の抽出結果

ごとの自己相互情報量を計算する．自己相互情報量は，単語がカテゴリに依存する度合いを示す量である [20]．その後，自己相互情報量が大きい順にソートし，上位 K 個を言及ありと分類したときの F1 値を計算する．F1 値は判定結果を評価するための尺度である [21]. $K = 1, 2, ...$ と変えていき，最もよい F1 値となる K を採用する．以上の過程で，たとえば位置情報リソースでは「GPS」等の単語が得られた．

前節と同様，単語ヒューリスティックの精度をシード値を 10 回変えた 10 分割の交差検証によって評価した．最終的な分類精度を表 4 に示す．全体に FPR は低いまま FNR が向上した．

5 ACODE による解析

本章では 20 万個のアプリに ACODE を適用した結果を示す． 5.1 節で実験に用いたデータセットについて述べる． 5.2 節で ACODE によってアプリを抽出した結果を述べ， 5.3 節でアプリが説明文で言及することなくプライバシー情報の取得を試みる原因を解明する．

5.1 データセット

本実験に用いた検体は公式マーケット [22] と 2 つのサードパーティマーケット [23, 24] から収集した．公式マーケットは説明文が英語のもの，サードパーティマーケットは説明文が中国語のものを対象とした．収集期間は 2012 年の 4 月から 2014 年の 4 月である．収集したアプリのうち，説明文が欠けている

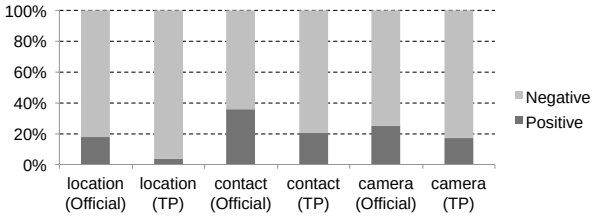

図 4: 各リソースを利用するアプリの言及と非言及の比率

ものやファイルが不完全なものを取り除いた後，無作為に 10 万個ずつ，計 20 万個のアプリを選択した．

5.2 抽出結果

図 3 にコード解析の結果を示す．全体として位置情報を取得するアプリが大きな割合を占めることがわかる．後述する通りこれは広告ライブラリの影響である．2 番目はカメラ，3 番目は連絡帳と続き，いずれのマーケットでも同じ順序であった．$C0$ と $C1$ の差分はパーミッションを過剰に要求しているアプリの数である．Felt ら [10] は，過剰要求の原因として開発者の不注意とインテント呼び出しを挙げている．実際インテントで利用されることが多いカメラリソースは過剰要求の割合が高い．インテントは受け取るアプリの設定によってパーミッションが必要な場合とそうでない場合があるが，いずれもアプリ自体はリソースを利用しない．また $C1$ と $C2$ の差分は API 呼び出しコードが到達不能となっているアプリの数である．このように実際にプライバシー情報にアクセスしないアプリは解析対象から除外する．

つづいて，図 3 の $C2$ に対してテキスト分類を適用した．この結果を図 4 に示す．説明文での言及があるものを positive，そうでないものを negative とした．公式マーケットにて公開されているアプリのうち，連絡帳，位置情報，カメラへのアクセスについて説明文で言及している割合は 17-35%であった．サードパーティマーケットにて公開されているアプリのうち，連絡帳，カメラへのアクセスについて説明文で言及している割合は 17-20%であり，位置情報ではわずか 4%であった．

5.3 原因の解明

アプリが説明文で言及せずにプライバシー情報を取得する原因を解明する．まず抽出された検体の傾向を分析し，その後具体的な原因を解明する．

5.3.1 抽出された検体の傾向

ACODE によって抽出された positive と negative の 2 つのクラスの差異を様々な観点から分析し比較を

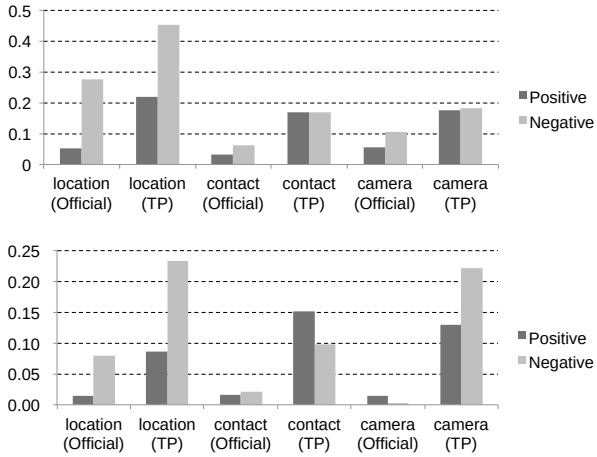

図 5: 抽出アプリにおけるアドウェアの割合（上）とマルウェアの割合（下）

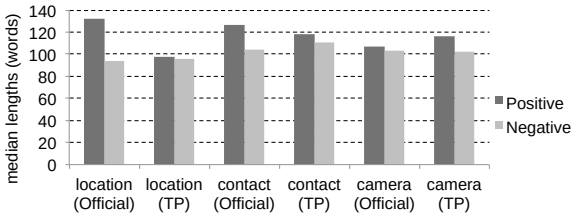

図 6: アプリの説明文に含まれる語数の中央値

行う．図 5 は検体をアンチウィルスソフトで解析した結果である．11 種類の商用ソフトウェアを利用し，1 つでも検知されたものはマルウェアおよびアドウェアとみなした．いくつかの例外を除きマルウェアやアドウェアと判定された割合は negative の方が高い．以上から，適切な説明文を持つアプリはマルウェアやアドウェアである可能性が低いといえる．

さらに我々は説明文のテキストの長さやアプリのクラス数，証明書ごとのアプリ数の分布を計測し，positive と negative の差異を見出した．図 6 に示す通り，negative の方が総じてテキストが短い．プライバシー情報へのアクセスについて言及していないアプリは，説明文自体が不足している傾向にあることがわかる．一方で図 7 に示す通りクラスの数は negative の方が多い．様々なリソースを活用する多機能なアプリは，すべての機能を説明文で言及しきれないためと考えられる．図 8 は上位 X%の証明書すなわち同一開発者によって開発されたアプリの数を示す累積分布関数のグラフである．この図から，言及なしでプライバシー情報へアクセスするアプリを少数の開発者が多数公開していることが読み取れる．たとえばサードパーティマーケットにおいて位置情報リソースを言及なく使用するアプリのうち 60%は，わずか 1%の開発者によって開発されている．

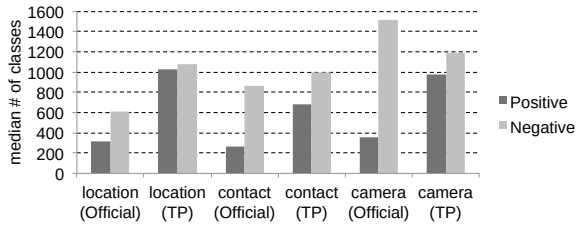

図 7: アプリに含まれるクラス数の中央値

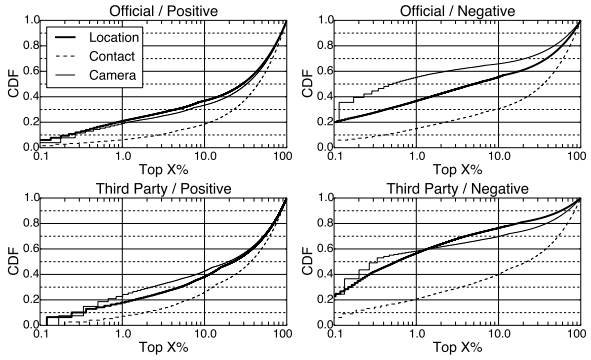

図 8: 上位 X % の開発者公開鍵証明書を持つアプリ数の割合

5.3.2 具体的な原因

本項ではアプリが説明文で言及せずプライバシー情報の取得を試みる具体的な原因を解明する．アプリの詳細な解析によって，3 つの主要な原因を見つけ出した．1 つ目はあえて言及されない副次的な機能によるもの，2 つ目はアプリ自動生成サービスによるもの，3 つ目は広告ライブラリによるものである．

- **言及されない機能**

説明文を注意深く観察したところ，アプリの副次的な機能によるリソースへのアクセスは言及されない傾向にあることがわかった．たとえば一部のゲームアプリでは，スコアを友人と共有するため連絡帳にアクセスする．このような機能はあくまでオプションであるため，説明文に記述されることは稀である．最も顕著な例として，カメラへのアクセスを要求する 2 次元コードリーダーが挙げられる．SNS やショッピングなど，数多くのアプリに 2 次元コードリーダーが付属しているが，主要な機能ではないためやはり説明文で言及されないことが多い．我々は 2 次元コードリーダーが及ぼす影響を測るため，ZXing [25] や ZBar [26] のような 2 次元コードリーダーライブラリを含むアプリの数を計測した．その結果カメラを言及せずに使用しているアプリのうち，公式マーケットでは 53%，サードパーティマーケットでは 66%が 2 次元コードリーダーライブラリを含んでいた．モバイルアプリマーケットは，副次的な機能によるプライバシー情報へのアクセスについても記述を求めるべきである．

- **アプリ自動生成サービス**

パッケージ名を分析したところ，アプリを自動的に生成するサービスによって作られたものが多数存在することがわかった．WEB 等で公開されている自動生成サービスを通じて，開発者はコードを記述することなくアプリを作成できる．ある 2 つのサービスによって作られたアプリ群を解析した結果，その全てが連絡帳，位置情報，カメラ等にアクセスするためのパーミッションと到達可能な API を含んでいた．この場合アプリがリソースへアクセスする機能を含むことを開発者が意図していないため，説明文に記述されることもない．公式マーケットにおいて連絡帳に言及なしでアクセスするアプリのうち，25%がこれらの自動生成サービスによって生成されたアプリであった．位置情報は 5%，カメラは 10%と，他のリソースにも影響を及ぼしていた．このサービスは中国では普及しておらず，サードパーティではこの傾向は見受けられなかった．

- **広告ライブラリ**

前項で示した通り，説明文とコードの間に齟齬があるアプリはアドウェアの割合が高い．この傾向は特に位置情報リソースで顕著である．一般に広告配信会社は地域と広告内容をマッチングさせるためにユーザーの位置情報を取得することがある [4]．広告と位置情報の相関性を調査するために，位置情報を取得する広告ライブラリを検出する方法を開発した．主要なアイディアとして，スマートフォンの広告通信をブロックするために集約された AdAway [27] 用の完全修飾ドメイン名 (FQDN) リストを用いる．アプリを逆アセンブルしたコードを解析し，位置情報を取得する API とリスト内の FQDN との共起関係を見る．ある FQDN が存在するとき必ず位置情報を取得する API が含まれる場合，そのような FQDN を位置情報取得 FQDN とした．調査の結果，説明文で言及せずに位置情報を取得するアプリのうち，公式マーケットでは 55%以上，サードパーティマーケットでは 78%以上が位置情報取得 FQDN を含んでいた．これらのアプリは開発者でなく広告会社の意向でリソースを使用するため，その旨が説明文に記述されない．モバイルアプリマーケットの提供者は，広告ライブラリによるプライバシー情報へのアクセスについて新たな通知手段を講じるべきである．

6 関連研究

本章では本研究と関連する研究を述べ，比較を行う．モバイルアプリの分野ではパーミッションやコードに基づく様々な研究が報告されてきたが，説明文に着目した例は少ない．Pandita ら [28] が提案する

表 5: 本研究と関連研究の比較

	ACODE	WHYPER	CHABADA
目的	コードと説明文の齟齬の分析	パーミッションの利用を示唆する文の特定	例外的な動作をするアプリ検出
検体数	200,000	581	32,308
マーケット	公式, TP	公式	公式
言語	英語,中国語	英語	英語
コード解析	コールグラフ解析	パーミッション確認	API解析
テキスト解析	BoW + SVM	意味解析	トピックモデル

WHYPER は，ユーザーの期待とアプリの実動作との隔たりを埋めることを試みる先駆的な研究である．WHYPER は自然言語処理によって説明文の意味を解析し，パーミッションについて言及しているセンテンスを抽出できることを実証した．Gorla ら [29] による CHABADA では，使用する API と説明文を特徴ベクトルとして教師なし学習で分類し，クラスタから外れ値を示す異常なアプリを特定する．一方 ACODE の目的は異常を検出することではなく，プライバシーの脅威をユーザーに通知する手段として説明文の有用性を定量化することである．そのために教師あり学習による二値分類を行い，リソースの使用を説明文が言及していないケースに焦点を当てる．さらにコールグラフ解析による API の到達性確認により，[28] および [29] と比較してセンシティブなリソースへのアクセスを精密に確認している．

本研究と上述した 2 つの研究との比較を表 5 に示す．技術的な違いや目的の違いに加え，本研究は実証規模の大きさでアドバンテージがある．

7 まとめ

公式マーケットとサードパーティマーケットから収集した 20 万のアプリに ACODE を適用し，説明文でプライバシー情報の取得について言及しているアプリは全体の約 4%-33%であることを解明した．本研究は開発者がプライバシー情報の取得を説明文で言及しないいくつかの理由を明らかにした．説明文の長さ，クラスの数，常に特定のリソースを使用する少数の開発者，言及されない副次的な機能，アプリ自動生成サービスによる不要な API の使用，広告ライブラリによるリソースの使用などである．今回は連絡帳，位置情報，カメラに限定して解析したが，ACODE を他のリソースに適用することで，各々が言及されない原因を明らかにできる．本研究は，モバイルアプリマーケットにおいてユーザーのプライバシー意識を向上させるための重要な足がかりとなる．リソースへのアクセスについて説明文で言及されない原因を意識することで，開発者はよりプライバシー情報に配慮した説明文を記述することが可能となる．モバイルアプリマーケットはプライバシー情報の取得を通知するための新たな手段として説明文を活用すべきであり，我々の研究成果はそのためのガイドラインとなることが期待される．

参考文献

[1] S. Shen and B. Blau, “Forecast: Mobile App Stores, Worldwide, 2013 Update.” https://www.gartner.com/doc/2584918/forecast-mobile-app-stores-worldwide.

[2] B. Report, “Pausing Google Play: More Than 100,000 Android Apps May Pose Security Risks.” https://www.bit9.com/research/pausing-google-play/.

[3] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, “Android permissions: user attention, comprehension, and behavior,” in *Proc. of SOUPS*, 2012.

[4] T. Book, A. Pridgen, and D. S. Wallach, “Longitudinal analysis of android ad library permissions,” in *Proc. of IEEE MoST*, 2013.

[5] E. Chin, A. P. Felt, V. Sekar, and D. Wagner, “Measuring user confidence in smartphone security and privacy,” in *Proc. of SOUPS*, 2012.

[6] Y. Zhou and X. Jiang, “Detecting passive content leaks and pollution in android applications,” in *Proc. of NDSS*, 2013.

[7] R. Templeman, Z. Rahman, D. Crandall, and A. Kapadia, “PlaceRaider: Virtual theft in physical spaces with smartphones,” in *Proc. of NDSS*, 2013.

[8] “Android asset packaging tool.” http://www.kandroid.org/guide/developing/tools/aapt.html.

[9] “Manifest.permission.” http://developer.android.com/reference/android/Manifest.permission.html.

[10] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, “Android permissions demystified,” in *Proc. of ACM CCS*, pp. 627–638, 2011.

[11] “PScout: Analyzing the Android Permission Specification.” http://pscout.csl.toronto.edu/.

[12] “android-apktool.” http://code.google.com/p/android-apktool/.

[13] “smali – An assembler/disassembler for Android’s dex format.” https://code.google.com/p/smali/.

[14] “androguard.” https://code.google.com/p/androguard/.

[15] “Kyoto Text Analysis Toolkit.” http://www.phontron.com/kytea/.

[16] “Natural Language Toolkit.” http://www.nltk.org.

[17] “imdict-chinese-analyzer.” https://code.google.com/p/imdict-chinese-analyzer/.

[18] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[19] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[20] K. W. Church and P. Hanks, “Word association norms, mutual information, and lexicography,” *Comput. Linguist.*, vol. 16-1, pp. 22–29, 1990.

[21] T. Sakai, “Metrics, statistics, tests,” in *Bridging Between Information Retrieval and Databases* (N. Ferro, ed.), vol. 8173 of *Lecture Notes in Computer Science*, pp. 116–163, Springer Berlin Heidelberg, 2014.

[22] “Google play.” http://play.google.com/.

[23] “Anzhi.com.” http://anzhi.com.

[24] “Nduoa market.” http://www.nduoa.com/.

[25] “Official ZXing (”Zebra Crossing”) project home.” https://github.com/zxing/zxing.

[26] “ZBar bar code reader.” http://zbar.sourceforge.net/.

[27] AdAway. http://adaway.org/hosts.txt.

[28] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, “Whyper: Towards automating risk assessment of mobile applications,” in *Proc. of USENIX Security*, pp. 527–542, 2013.

[29] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, “Checking app behavior against app descriptions,” in *Proc. of ICSE*, 2014.