メタデータを用いたリパッケージアプリの検出手法「CloneSpot」の改良

藤　竜成 [a)]・五味　健一郎 [b)]・油田　健太郎 [c)]・山場　久昭 [d)]・岡崎　直宣 [e)]

An Improved Method of "CloneSpot" for Detecting Repackaged Applications Using Metadata

Ryusei FUJI, Ken'ichiro GOMI, Kentaro ABURADA, Hisaaki YAMABA, Naonobu OKAZAKI

Abstract

Since Android is widely used as an operating system (OS) for smartphones, applications running on the OS are being targeted by malware developers. Android applications are generally created using Java, which means it is easy to decompile the application and add or change the code; therefore, malware developers can easily insert malware and illegal advertisements into legitimate applications. The modified applications are called repackaged applications. These applications are uploaded to a market for users to install, however since repackaged applications have a negative effect on users, their detection and removal are important. The Developers of repackaged applications tend to be reluctant to make large changes to data, called application metadata, to make users misrecognition. To detect repackaged applications, "CloneSpot" has been proposed which uses the above trends, however, improvements are required for efficient detection. In this study, we propose an improved method of "CloneSpot" for the efficient detection of repackaged applications. Specifically, we improve the clustering of "CloneSpot" and introduce evaluation indexes for efficient repackaged application detection. In the evaluation experiments, we evaluated the clustering accuracy and the repackaged application estimation accuracy of the proposed method.

Keywords: Android, repackaged application, metadata, MinHash

1. はじめに

最新の推定によると、世界のスマートフォン利用者数は 2018 年末に 33 億人に達し、2021 年までに 38 億人まで増加すると予測されている [1)]。その中で Android は、スマートフォン業界の主要なオペレーティングシステムとして位置付けられており、2019 年に Android ユーザー数は 25 億人に及んだ [2)]。このような人気により、Android はマルウェア開発者から狙われるようになった。現在、GooglePlay[3)] で利用可能な全アプリケーションの約 3〜4 %がマルウェアであると推定されている [4)]。

Android アプリケーションの多くは、Java で作られている。Java はリバースエンジニアリングが可能であるため、簡単にアプリケーションコードを逆コンパイル、コードの追加や変更、マーケット上に再アップロードすることができる。この流れを、Android アプリケーションのリパッケージと呼び、正規のアプリケーションに対して、マルウェアを施したり、広告の追加や変更を施すための手法の 1 つとなっている。

リパッケージの対策として、アプリケーションコードの類似性やアプリケーションの実行パターンの調査による検出などがある。その一方で、Ignacio Martín らは、メタデータを使ってリパッケージアプリを検出をする手法「CloneSpot」を提案した [5)]。メタデータとは、アプリケーションの「タイトル」や「説明」などに当たる情報のことで、プログラムとは違い、GooglePlay などのマーケット上で簡単に入手できる。リパッケージアプリの開発者は、ユーザーに正規のアプリケーションと認識させるために、メタデータに対し、大きな変更を加えることへ消極的な傾向がある。「CloneSpot」はその特徴を使って、クラスタリングによる類似アプリケーションのグループ化と各グループのランク付けを行う手法であり、コードの類似性や実行パターンによる手法よりも、高速に検出することが期待できる.

本論文では、提案された手法に対し、各グループのランク付けの際に、関係性が薄いアプリケーションを削除することで、ランク付けの精度向上を目指す。また、メタデータ「評価数」を用いて、アプリケーションの評価をすることで、リパッケージアプリの推定を行い、手動による評価の時間短縮を目指す。

以下、本概要の構成を述べる。第 2 章では先行研究の「CloneSpot」について述べ、問題点を指摘する。第 3 章では提案手法について述べ、第 4 章では実験を行った結果と考察を示す。第 5 章ではまとめと今後の課題について述べる。

2. 先行研究

Ignacio Martín らの研究で提案された「CloneSpot」の手順と問題点を以下に示す [5)]。

[a)]工学専攻機械・情報系コース大学院生
[b)]情報システム工学科学部生
[c)]情報システム工学科准教授
[d)]情報システム工学科助教
[e)]情報システム工学科教授

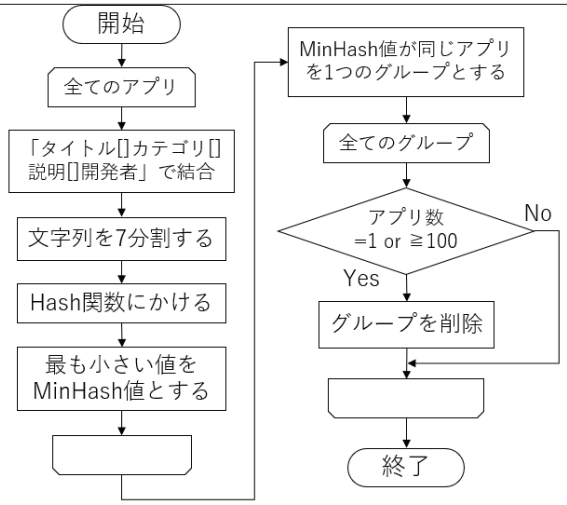

図 1. クラスタリングの流れ

2.1 「CloneSpot」の手順

「CloneSpot」は大量のアプリケーションを MinHash と呼ばれる手法でクラスタリングを行い、クラスタリングしたグループにランク付けを行う。その後、ランクが高い順に手動でアプリケーションを評価することで、リパッケージアプリを検出する。

クラスタリングを行う目的は、オリジナルアプリとリパッケージアプリをひとまとまりにすることである。クラスタリングには MinHash が使われ、その流れを説明する。初めに、全てのアプリケーションに対して、メタデータ「タイトル」「カテゴリ」「説明」および「開発者名」を結合し、1 つの文字列とする。その際、空白を区切り文字とし、その後、結合された文字列を 7 文字ずつ分割する。次に Hash 関数を用いて、分割された文字列全てに対し、ハッシュ値を求め、その中で最小の値を MinHash 値とする。最後に MinHash 値が同じアプリケーションを 1 つのグループとして集約する。その際、各グループのアプリケーション数が 1 または 100 以上の場合、そのグループを削除する。以上でクラスタリングは完了する（図 1)。

クラスタリングの後は、グループのランク付けを行う。ランク付けの目的は、各グループのアプリケーションの類似性を数値化することであり、メタデータ「タイトル」と「説明」を利用する。クラスタリングされた各グループにおいて、アプリケーションの総当たりで「タイトル」の編集距離と「説明」のコサイン類似度を求め、ASI(Application Similarity Index) を算出する。ASI は以下の式で表す。

$$ASI(a、b) = \frac{median(\vec{CS}(a、b))}{\varepsilon\ (\vec{ED}(a、b))}$$

アプリケーション a、b に対して、$\vec{CS}$ と $\vec{ED}$ はコサイン類似度と編集距離を意味する。また、$median$ と ε は、それぞれ中央値と平均を意味する。ASI の値が高いほど、グループ内のアプリケーションのメタデータは類似していると言える。各グループで ASI を算出した後、ASI が降順になるように、グループをソートすることで、グループのランク付けは完了する（図 2)。

以上が「CloneSpot」の流れで、リパッケージアプリの最終的な評価は手動で行う。ASI の値が高い順に検査することで、効率よく評価を行うことができる。

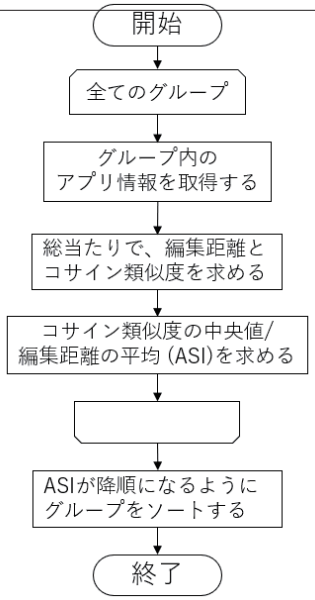

図 2. グループのランク付けの流れ

2.2 問題点

「CloneSpot」はクラスタリングの手段として、MinHash が使われている。MinHash は分割された文字列を元に Hash 値を求めているので、似たアプリケーションのメタデータでクラスタリングすることができる。しかし、稀に関係の薄いアプリケーションが混ざり、ASI の値を下げることがある。その結果、早い段階で評価して貰えるはずのグループが遅くなってしまい、手動による評価の効率が下がる。

また、最終的なアプリケーションの評価は手動で行われる。各グループには、類似したアプリケーションが存在するため、指標もなしに、正規アプリとリパッケージアプリを見分けるには時間や手間がかかる。

3. 提案方式

本研究では、「CloneSpot」に対し、2 点の改良を提案し、グループのランク付けの精度と手動による評価の時間短縮を目的とする。

3.1 クラスタリングの改良

2.2 節で述べたように、クラスタリングを行う際、関係の薄いアプリケーションが混ざることがある。これを解消するためには、関係の薄いアプリケーションをグループ内から削除する必要がある。

提案する「CloneSpot」の改良手順を図 3 に示す。図 2 のグループのランク付けの流れがベースになっている。全てのグループに対し、アプリケーションの「タイトル」と「説明」のメタデータを取得し、取得した順番に番号を割り振る。次にアプリケーションの総当たりで「タイトル」の編集距離、「説明」のコサイン類似度を計算する。編集距離が 3 より小さいかつ、コサイン類似が 0.5 以上のアプリケーションの組み合わせが存在した場合、アプリケーションの番号をリストに追加する。その後、リストに番号が存在しているとき、リスト内のアプリ番号の総当たりで ASI を求め、存在していないと

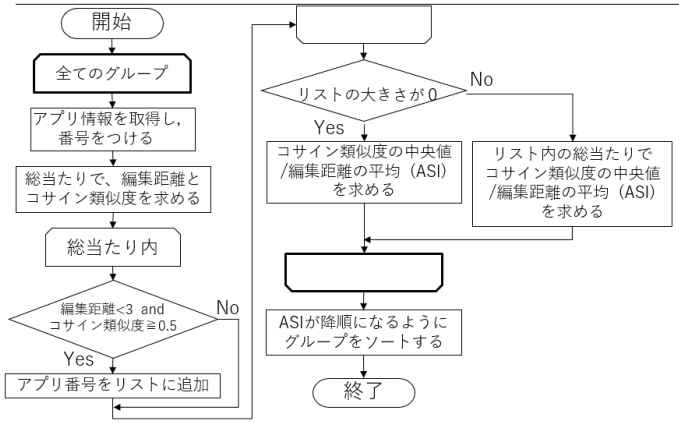

図 3. 改良したグループのランク付けの流れ

表 1. アプリケーションの評価

オリジナルアプリとリパッケージアプリどちらの可能性もある	△
リパッケージアプリの可能性が高い	×

きは、先行研究通りに ASI を求める。最後に、ASI が降順になるようにグループをソートすることで終了する。

リストには、クラスタリングされたときより、さらに似たアプリケーションの番号が入る。リスト内に番号が存在するとき、リスト内にないアプリケーション、つまり、関係が薄いアプリケーションを削除することができる。

3.2 リパッケージアプリの推定

2.2 節で述べたように、「CloneSpot」はリパッケージアプリの評価を最終的には手動で行う。その手間を解消するため、メタデータ「評価数」を使ったリパッケージアプリの推定を提案する。

提案する「CloneSpot」の改良手順を図 4 に示す。図 4 は「星 1 の数」でリパッケージアプリを推定する際のフローチャートである。前提として、1 つのグループに対して、1 つのオリジナルアプリが存在し、他のアプリはリパッケージアプリと考える。ここでは、オリジナルアプリの検出よりも、リパッケージアプリの検出を優先するため、評価方法を以下の表 1 に示す。各グループに対して、アプリケーションの情報を取得した後、メタデータ「星 1 の数」で比較を行う。ここでは「星 1 の数」で比較を行うので、評価数が最小のアプリケーションが「△」となり、それ以外は「×」となる。評価数が最小のアプリケーションが 2 つ以上出てきた場合は、どちらとも「△」とし、それ以外のアプリケーションを「×」とする。

Alfonso Munoz らの研究で、「評価数」を用いたマルウェアアプリの検出は非常に困難であることが判明している [4]。よって、リパッケージアプリの検出においても、「評価数」を用いた検出は極めて困難であると考える。しかし、クラスタリングであらかじめ似たアプリケーションがひとまとまりになっているため、クラスタリングが行われていない場合と比較すると有効ではないかと考える。ここでは、「星 1 の数」だけではなく、「星 1 の割合」や、「星 5 の数」や「総評価数」などを使って、リパッケージアプリの判定を行う指標を作る。その際、図 4 の最小評価数は指標によって、最大評価数になる場合がある。

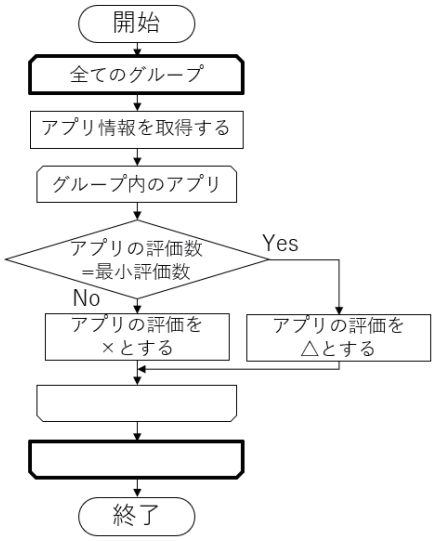

図 4. アプリケーションの評価の流れ

表 2. 実験に使用したデータ

アプリケーション数	1,379,861
収集時期	2017 年 9 月
メタデータ	タイトル 説明 カテゴリ 開発者 評価数

4. 実験と考察

提案方式のグループのランク付けの精度とリパッケージアプリの推定精度に関して、評価実験を行い、結果に対しての考察を行う。

4.1 実験データ

実験に使用したアプリケーションのメタデータは、先行研究と同一のデータセットを使用した。データセット内には、メタデータ取得日など、今回の実験に不要なデータも存在した。表 2 にデートセット内で実験に使用したデータを示す。「評価数」は星 1 から星 5 までのそれぞれ数と、「総評価数」を表す [6]。

先行研究のデータセットの中には、リパッケージアプリか否かの判定が存在しなかった。よって、アプリケーションの URL にアクセスして、GooglePlay 内にアプリケーションの情報が存在するかどうかで判定した。2019 年 12 月にアクセスし、データを収集した [3]。

4.2 前処理

提案方式の実験を行う前に、アプリケーションに対してのクラスタリングを行う必要があり、図 3 に当てはまる。クラスタリングされた後のアプリケーション数とグループ数を表 3 に示す。初期のアプリケーション数は 1,379,861 個もあったが、ほとんどグループは、アプリ数が 1 だったため、グループが削除された。また、アプリ数が 100 以上のグループは 1,217

表 3. クラスタリング後のデータ

アプリケーション数	257,641
グループ数	64,272

表 4. 改良による、ASI ごとのグループ数の変化

ASI	改良前	改良後	差
0.0〜0.1	65,043	64,527	- 516
0.1〜0.2	270	255	- 15
0.2〜0.3	74	86	+ 12
0.3〜0.4	39	55	+ 16
0.4〜0.5	43	70	+ 26
0.5〜0.6	22	112	+ 90
0.6〜0.7	16	77	+ 61
0.7〜0.8	23	81	+ 58
0.8〜0.9	26	85	+ 59
0.9〜1.0	145	353	+ 208

表 5. 改良による、ASI ごとのアプリ数の変化

ASI	改良前	改良後	差
0.0〜0.1	256,246	251,036	- 5,210
0.1〜0.2	592	543	- 49
0.2〜0.3	157	178	+ 21
0.3〜0.4	81	118	+ 37
0.4〜0.5	86	150	+ 64
0.5〜0.6	47	246	+ 199
0.6〜0.7	32	160	+ 128
0.7〜0.8	47	174	+ 127
0.8〜0.9	56	182	+ 126
0.9〜1.0	297	722	+ 425

個あり、同様に削除された。

4.3 クラスタリングの改良結果

改良前と改良後の ASI ごとのグループ数とアプリケーション数の変化を表 4 と表 5 に示す。ASI の値は高いほど、グループ内のアプリケーションのメタデータが類似しているといえる。ASI が 0.0〜0.1、0.1〜0.2 のグループが 0.2 以上のグループに分布する結果となった。それに伴い、アプリケーションも ASI が 0.0〜0.1、0.1〜0.2 から 0.2 以上に分布した。また、改良前のアプリケーション数が 257,641 個だったのに対し、関係の薄いアプリケーションが削除されたことにより、253,509 個と 4,132 個減少した。グループ数の差を見ると、ASI が 0.0〜0.1 のグループが最も減少し、0.9〜1.0 のグループが最も増加した。これは、グループ内のほとんどのアプリケーションのメタデータが似ていたが、1 つのアプリケーションのせいで、異常に ASI を下げていたと考えられる。また、0.2〜0.3 や 0.4〜0.5 にも分布した理由として、ASI の計算方法に問題があると考える。ASI の計算は、「説明」のコサイン類似度よりも、「タイトル」の編集距離に大きく依存しており、提案方式のクラスタリングで一定のアプリケーションは削除されたものの、消されていないアプリケーションの中に編集距離が高いものが存在し、ASI の値を低くしたと考えられる。

表 6. リパッケージアプリ推定の評価

評価指標	Precision	Recall	F1-score
星 1 の数	0.65	0.33	0.44
星 1 の割合	0.68	0.35	0.46
星 5 の数	0.79	0.51	0.62
総投票数	0.78	0.51	0.62
星 5-(星 1+星 2)	0.79	0.52	0.63

4.4 リパッケージアプリの推定結果

リパッケージアプリ推定の評価を表 6 に示す。ここでは、ASI が 0.9 以上のアプリケーションで推定を行う。理由は、ASI の値が小さい場合、アプリケーションが類似していない可能性があるためである。Precision は「×」と評価して、実際に GooglePlay に情報が存在しなかった割合を表し、Recall は GooglePlay に情報が存在しないアプリケーションのうち、「×」と評価できた割合を表す。F1-score は Precison と Recall を総合的に評価する指標である。表 6 から「星 1 の数」や「星 1 の割合」よりも、「星 5 の数」や「総評価数」の方が、全ての精度で上昇した。また、改良と検証を重ね、精度が 1 番良くなった指標は「星 5-(星 1+星 2)」であった。星 5 を信頼値とし、星 1 と星 2 の合計を不信値とみなすことで、精度が上昇したのではないかと考える。しかし、Recall が 0.5 台までにしかならなかった。理由として、アプリケーションとリパッケージ判定のデータセットの取得時期の差にあると考える。取得時期の差は約 2 年で、評価が高いアプリケーションでも、サービス終了などで GooglePlay に存在しないものがあった。また、Precision も 8 割に満たなかった。理由として、リパッケージアプリと判定したアプリがオリジナルアプリの関連アプリだった可能性がある。また、意図的に評価を下げられた可能性も考えらる。

5. まとめ

本研究では、メタデータを用いたリパッケージアプリ検出手法「CloneSpot」に対して、2 点の改良を提案した。1 点は、クラスタリングされたグループ内において、関係の薄いアプリケーションを削除することで、ランク付け精度の向上を目指した。結果、ASI の値が低いグループは、関係の薄いアプリケーションを削除することで、ASI の値を高くすることができた。もう 1 点は、メタデータ「評価数」を用いてリパッケージアプリを推定することで、手動による評価の時間短縮を目指した。結果、事前にクラスタリングを行うことで、リパッケージアプリの推定に対する一定の効果があった。

今後の課題として、Precision、Recall 及び F1-score の更なる向上が必要である。そのために「評価数」以外のメタデータで検証したり、最新のデータセットを用いて検証していかなければならない。また、ASI の精度向上のために、グループのランク付けをベースとする改良を行ったが、MinHash 以外のクラスタリングによるグループ化の精度向上が見込めないか検証し比較を行いたい。

参考文献

1) http://jp.xinhuanet.com/2018-09/14/c_137468242.htm, (2020-1-22 閲覧).
2) https://jp.techcrunch.com/2019/05/09/2019-05-07-android-now-has-2-5b-users/, (2020-1-22 閲覧).
3) https://play.google.com/store?hl=ja,(2020-1-22 参照).
4) A Munoz, et al.: Android malware detection from Google Play meta-data: Selection of important features, 2015 IEEE Conference on Communications and Network Security (CNS), pp. 701-702, 2015.
5) I Martín and JA Hernández: CloneSpot: Fast detection of Android repackages, Future Generation Computer Systems 94, pp.740–748, 2019.
6) https://data.mendeley.com/datasets/2n595bbm3v/4, (2020-1-22 閲覧).