The ISC Int'l Journal of Information Security

ISeCure

October 20225, Volume 17, Number 3 (pp. 1–8)

http://www.isecure-journal.org

Presented at the ISCISC'2025 in TEHRAN, Iran.

GP-FACL: A Dataset of FAlse CLaim Descriptions and Functionalities of Google Play Apps **

Sepehr Mehregan [1,*], Mahdi Tamjidi [1], Amir Hossein Rahimi [2], Ava Razavi [3], Sadegh Eskandari [1], and Seyed Amir Hossein Tabatabaei [1]

[1] *Department of Computer Science, University of Guilan, Rasht, Iran*
[2] *Department of Computer Science, Aston University, Birmingham, United Kingdom*
[3] *Department of Computer Science, Brock University St. Catharines, Canada*

ARTICLE INFO.

Keywords:
Automated Collection, Dataset Creation, Fraud Detection, Mobile Applications, Textual Deception

Type:

doi:

Abstract

Mobile phones are among the most significant technological advancements, offering unmatched convenience and seamlessly integrating into modern lifestyles. However, their widespread use also facilitates both beneficial and harmful practices. The absence of comprehensive datasets with reliable app descriptions undermines user confidence in Google Play as a trustworthy platform for software. To address this gap, we introduce a new dataset, GP-FACL, in this study. This dataset contains "fake apps" that make false claims in their descriptions and pretend to offer features that do not actually exist. Applications were first manually collected, after which keywords were extracted to generate 2-gram key phrases. These key phrases were then used to automate the collection of additional applications. The final dataset provides a systematic approach for identifying false-claim applications across a variety of app categories. Our approach resulted in 117 applications being verified as containing erroneous or misleading claims. This dataset offers researchers and practitioners a valuable resource for advancing fraud detection and mitigating deceptive applications on mobile platforms.

1 Introduction

Mobile phones have transformed communication because of their widespread use. Along with the rise in mobile phone usage, mobile applications, also known as apps, have seen significant growth in development. The number of mobile applications available in the Google Play Store has risen by 2.4 million since 2009. Consequently, the risk of encountering fraudulent applications designed to exploit or harm users has also increased [1]. Such apps often disappoint users by overstating their features or benefits. Although not always overtly harmful, these deceptive apps can waste users' time, display misleading advertisements, or even lead to identity theft, privacy violations, and

* Corresponding author.
**The ISCISC'2025 program committee effort is highly acknowledged for reviewing this paper.
Email addresses: sepehrmehreghan@gmail.com, mahditamjidi4@gmail.com, 240142155@aston.ac.uk, ua23vc@brocku.ca, eskandari@guilan.ac.ir, amirhossein.tabatabaei@guilan.ac.ir
ISSN: 2008-2045

financial loss. Users should be aware of the dangers of downloading apps with false descriptions and be careful when downloading apps from developers they do not know or trust. App descriptions are a primary tool for users to evaluate an app's suitability for their needs. Descriptions provide important details that help users assess whether an app meets their needs and expectations. Nevertheless, some developers violate this trust by developing "fake apps" with misleading descriptions. These apps deceive users by advertising features they do not actually have (e.g., claiming to work offline while requiring an internet connection), primarily to increase downloads or ad revenue. Unlike malware, they usually do not contain malicious code intended to steal data or damage devices; they exploit trust for profit rather than harm. It is essential to identify and address such fraudulent apps to protect users' privacy and maintain the integrity of the mobile app ecosystem.

To address this problem, we present GP-FACL, a dataset explicitly created to identify falsely claiming fake Google Play apps. We have gathered 1,026 suspicious apps using a systematic approach (described in the methodology section) and confirmed 117 of them as real fake apps. In contrast to previous datasets, which focus on detecting malware, ad fraud, or counterfeit apps, GP-FACL is the first dataset specifically targeting textual deception in app descriptions, an important yet understudied form of mobile app fraud. This dataset provides an organized collection of metadata (app descriptions, developer details, installation counts, and user reviews), enabling researchers to build stronger fraud-detection models. Through GP-FACL, we fill the critical gap in structured resources for deceptive apps and provide a valuable benchmark dataset to improve the detection of fraudulent mobile applications in app marketplaces.

The paper discusses the problem of fraudulent mobile app descriptions in marketplaces by constructing and examining the GP-FACL dataset. Section 2 gives a brief description of the previous studies on mobile app fraud detection to put our contribution into perspective. Section 3 explains the semi-automated method for creating GP-FACL, including the app collection and filtration procedures. Section 4 describes the structure and attributes of the dataset, and its main findings. Section 5 addresses the difficulties that were faced in the course of research. Lastly, Section 6 concludes the work on GP-FACL and discusses how it can be used in future research on app store security.

2 Related Work

Fraudulent and deceptive applications in mobile app marketplaces are a relatively unexplored field of detection. Despite the release of some datasets and research works, the majority of them are domain-specific, addressing such issues as malicious app detection, ad fraud, or fake applications. None directly addresses the issue of fake app descriptions, where apps claim to perform functions they do not provide. Here, we review the literature on the topic of our study and outline the limitations of previous datasets that motivated the development of GP-FACL. One of the first malware datasets of Android applications is AndroZoo, proposed by Allix *et al.* [2], which was gathered across several marketplaces, including Google Play. Although AndroZoo offers a vast set of apps with diverse behaviors, it primarily enables code-level analysis and the detection of behavioural anomalies via labels from multiple antivirus engines.

Besides AndroZoo, other studies, such as MAd-Fraud by Crussell *et al.* [3], have been dedicated to detecting ad fraud in Android apps. Mad Fraud detects and blocks apps that commit ad fraud (fake requests, impressions, or clicks) without genuine user interaction, using techniques like emulators, botnets, click injection, SDK spoofing, and traffic manipulation. The dataset includes over 130,000 apps from various marketplaces and provides useful insights into ad fraud behaviour. Nevertheless, the study is limited to ad fraud and does not address broader fraudulent activities, such as false app descriptions.

Rahman *et al.* proposed FairPlay [4], a system that detects suspicious apps through correlating review activities and integrating linguistic and behavioural signals. This method achieves over 95 % accuracy in classifying applications as malicious, fraudulent, or benign. Although FairPlay shows good results, it is strongly dependent on user reviews, which are very noisy and unreliable in linguistic tasks. Thus, it is ineffective at identifying fraudulent claims in app descriptions.

Muppavaram *et al.* [5] examined various types of fake apps in the Android ecosystem and classified them based on similarities in icons, names, and functionality with legitimate apps. They introduced the Fake App Detection (FAD) technique, which leverages certificate attributes and permissions to distinguish between fake and legitimate apps, achieving 96.9% accuracy on a set of 574 apps (274 fake and 300 legitimate). Although their work proposes a static analysis method for detecting fake apps, it does not specifically target textual deception in app descriptions. It relies on features that may fail to capture misleading functionality claims.

Rani *et al.* [6] designed a fake app detector based on sentiment analysis of user reviews. They do this by gathering and pre-processing reviews, sentiment analysis using TextBlob, and classifying apps as fraudu-

lent or legitimate using logistic regression. The system detects fraud based on negative sentiment patterns on apps such as Flipkart (legitimate) and Auto Answer Call (fake). However, it relies on the quality and authenticity of user reviews, which can be faked. It fails to confirm the accuracy of how apps are described and how they actually work.

The shortcomings of the available datasets underscore the need for a more comprehensive resource to combat fraudulent claims in mobile app marketplaces. The majority of previous research either focuses on a specific area or lacks the depth needed to assess the validity of app descriptions. Moreover, not many of them combine practical testing with an evaluation of the apps' actual functionality. To fill this gap, we have created the GP-FACL dataset, which focuses on fraudulent claims in mobile applications on a wider scope. In contrast to earlier datasets, GP-FACL consists of applications whose advertised features have been manually verified through practical testing, making the dataset significantly more reliable.

3 Methodology

While generating the GP-FACL dataset, we found that manually identifying fake apps was extremely time-consuming and inefficient. Therefore, we had to develop a semi-automatic procedure to detect them. Our method (Figure 1) consisted of three steps: manual app collection, automatic app collection, and filtration. To begin with, we manually identified a group of known fake apps and took them as a starting point. Next, we identified the best keywords from the app descriptions and searched for them on Google Play. This resulted in the collection of 1,026 apps. Next, we used the app, filtering by certain features, to reduce the list to the ones more likely to be fake. Lastly, the most suspicious apps were installed manually and checked to ensure that they were fake. This led to the discovery of more than 117 fake apps. The specifics of each step are described in the following subsections.

3.1 Manual App Collection

Our dataset was built from a manual app collection phase, which enabled us to identify an initial set of fake apps. The research team consisted of six people, including two senior researchers, all of whom had a computer science background and experience with mobile app ecosystems, and manually searched and assessed potentially fraudulent apps on the Google Play Store. To cover all app categories in the store, we assigned the four team members to these categories, with each member exploring at least two. The two supervisors ensured consistency in evaluation criteria and resolved any disagreements during the review process. We looked at the apps on the lower end of each category, which we defined as apps with less than 1,000 downloads or not in the top 100 in their category listing, as it was hypothesised that the apps in this range were more likely to have fraudulent features because they were less visible and possibly less carefully developed.

The identification process involved searching for apps with suspicious characteristics, such as poor descriptions, inconsistent ratings, or generic icons. Android emulators were used to download and install suspicious apps in a sandboxed environment to reduce the risk of potential malware. The functionality of each app was thoroughly tested to ensure it is consistent with the description provided on the Google Play Store. For example, an app marketed as a professional video editor was tested for its advertised editing capabilities and the availability of promised features. Apps that did not perform as advertised were considered fake.

In this manual process, we have created a dataset of 29 fake apps. The data contained important information about each application, including its Google Play Store URL, app ID, assigned category, official description, and a comprehensive explanation of why it was classified as fake. The most common reasons to declare an app as fake were the lack of functionality of core features, false descriptions, or data harvesting detected during the initial analysis of network traffic. To maintain consistency, every flagged app was reviewed by at least 2 team members independently, and any disagreements were discussed. This manually curated dataset served as the foundation for the automated collection process that followed and provided useful insights into the nature of fake apps.

3.2 Automatic App Collection

To expand our dataset beyond the initial 29 fake apps, we employed an automated method to collect a larger set of potentially suspicious apps. Our method targets textual deception, where fraud lies in the app's description. We hypothesised that developers reuse deceptive terminology across apps to attract users, creating semantic patterns. Given the limited set of plausible but false claims per app category, misleading phrases are likely reused. Thus, using text similarity to extract keywords from the descriptions of known fake apps is a direct way to identify others with similar misleading language. We used KeyBERT [7], a powerful keyword-extraction model based on BERT embeddings, to analyze the descriptions of the 29 fake apps. We chose KeyBERT over other keyword extraction algorithms, including YAKE, RAKE, and TextRank, because it offers higher coverage of 60%,

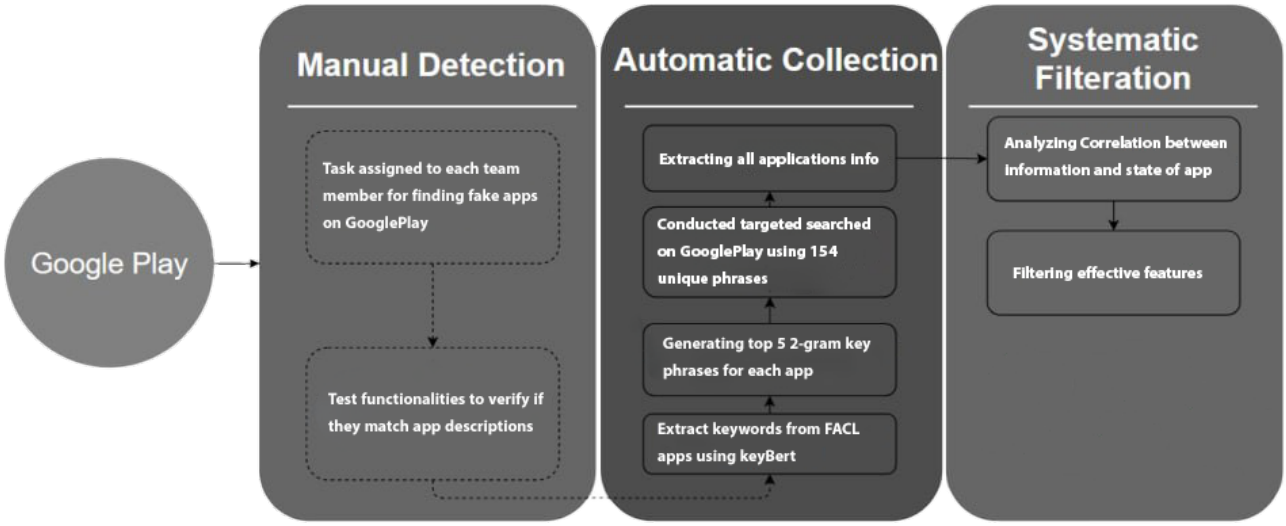


Figure 1. Approach Protocol for GP-FACL Dataset Creation

precision of 60%, and an F1-score of up to 64% on benchmark datasets [8], with an average F1-score of 39.43% across various benchmark datasets [9]. KeyBERT's BERT-based approach captures semantic relationships in text, making it suitable for extracting contextually relevant 2-gram key phrases (e.g., "photo editor," "free game") that reflect deceptive language. We selected 2-gram phrases because they produce more relevant search results than unigrams or higher-order n-grams, balancing specificity and generalizability. This process yielded 154 unique 2-gram key phrases after eliminating duplicates.

These search terms were applied to conduct specific searches on the web version of the Google Play Store. Due to platform limitations, no more than 30 apps were displayed per search query. To overcome this and gather comprehensive data, we used the Google Play Scraper [10] library to programmatically scrape app information, including title, URL, developer, user ratings, number of reviews, estimated downloads, and description. We also configured the scraper to retrieve other apps published by the same developers as those in the search results, as fake apps are frequently developed by a few developers who run multiple fraudulent apps. This procedure yielded a list of 1,026 suspicious apps, including metadata such as reviews, installs, ratings, and developer information. The effectiveness of this text-similarity approach is demonstrated by the successful identification of 1,026 suspicious apps, of which 117 were confirmed as fake after manual verification (Section 3.3), achieving a precision of approximately 85–90%. This expanded dataset provided a broad range of information for the next filtering stage, confirming the effectiveness of the text-similarity approach in targeting apps likely to contain false claims.

3.3 Filtration

During the filtration stage, our primary goal was to make it easier to identify the actual fake apps among the 1,026 apps in our dataset. To achieve this, we established criteria to filter the dataset. Our analysis of identified fake apps helped us find common factors that could assist in the filtration process. Due to Google Play policies, metrics such as the number of reviews and ratings would not remain consistent across devices. Therefore, they could not be trusted for the filtration process. For example, Google Play may show a 3.2 rating for a certain app on the mobile version and a 4.1 rating for the same app on the web version. We needed to find features that remained consistent across devices. To address this, we decided to implement three flags, each assigned a value of 1 if triggered and zero if not.

The first flag focused on whether the developer had registered an address. Our research revealed that most fake apps lacked a registered developer address. The second flag, the update time flag, indicated whether an app had been abandoned. If the release time and update time of an app were the same — indicating the app had never been updated — this flag would be triggered. For the third flag, we focused on the app's website. We observed that many fake apps used subdomains like ".blogspot.com" instead of traditional domain endings like ".com". These subdomains were often free and even offered affordable app development features. Additionally, we considered the registered email address. This flag would be triggered if the website address ended in a subdomain AND the subdomain of the website differed from the domain of the email address. For example, if an app used a " .blogspot.com " address but had a corresponding email address ending with "@gmail.com".

To further improve the dataset, we removed apps that had more than 10,000 downloads. This step ensured that widely used authentic apps were not included in the database. We added the apps that had fewer than 10,000 downloads and triggered either the update time flag OR both the address AND website flags simultaneously. Finally, a dataset of 1,026 apps was retained as our benchmark.

3.4 Manual App Analysis

The last step was to manually verify the 504 filtered apps to ensure that they were fake. Manual analysis was necessary because there are no reliable automated methods to identify inconsistencies between an app's description and its functionality. To prioritize the most suspicious apps, we initially considered those that raised all three flags (address, update time, and website), resulting in 181 apps. Then we analyzed apps that only set the update time flag, and added 323 apps to the list of apps to analyze. We also added apps published by the same developers as the flagged apps, even when they did not meet the flag criteria, to account for the possibility that developers may publish multiple fake apps.

All the apps were downloaded and installed on a secure Android emulator, and their functionality was tested against their description on the Google Play Store. Fake apps were categorized as such when they failed to deliver the advertised features. To ensure accuracy, at least two team members assessed every app, and disagreements were resolved by consensus. This procedure identified 88 additional fake apps, bringing the total to 117 (29 in the manual collection phase and 88 in the filtered dataset). The resulting GP-FACL Dataset contained comprehensive metadata for each fake app, including its URL, app ID, category, description, and rationale for classification, providing a solid basis for further analysis, including pattern identification and the creation of automated detection tools.

4 Dataset Characterization and Analysis

The GP-FACL dataset, comprising 1,026 records and 20 attributes, provides rich metadata on mobile apps from the Google Play Store to detect deceptive apps. This section describes the dataset's structure, major attributes, statistical summary, and an analysis of the 117 confirmed fake apps, including common false claims, flag correlations, and filtering method performance.

4.1 Dataset Structure and Attributes

The dataset consists of 1,026 applications, 117 of which are verified as fake using the methodology in Section 3. The entries contain 25 attributes that describe the app's metadata, the developer, and user interaction. Key attributes are:

- **App Identifiers**: `Appid`, `App Title`, `Summary`, and `Description`, capturing unique identifiers and textual content for app identification and analysis.
- **Temporal Data**: `ReleaseTime` and `UpdateTime`, recording the app's launch and last update dates to assess maintenance patterns.
- **User Engagement Metrics**: `Score` (user ratings, 0–5), `TotalReviews` (number of reviews), and `Realinstalls` (installation counts), reflecting app popularity and user feedback.
- **Developer Details**: `DevWebsite`, `DevEmail`, and `DevAddress`, frequently incomplete or suspicious in deceptive apps, aiding fraud detection.
- **Detection Flags**: `DevWebsiteFlag`, `DevAddressFlag`, `UpdateTimeFlag`, and `RealinstallsFlag`, binary indicators flagging suspicious traits, such as absent addresses or identical release/update dates.

Furthermore, the dataset is accompanied by a JSON file containing user reviews of each app. Metadata in these reviews includes the app version, the review date, the user rating, and the developer's responses. An example of a review in the JSON format is given below:

Listing 1: Example JSON-formatted user review


```
{
  "appVersion": "1.0.0",
  "at": "2023-08-26 21:42:18",
  "content": "I. verify your number",
  "repliedAt": null,
  "replyContent": null,
  "reviewCreatedVersion": "1.0.0",
  "reviewId": "bb0668e8-e1b1-4b6f-ab7f-5dc7cd49ac8f
",
  "score": 3,
  "thumbsUpCount": 1,
  "userImage": "https://play-lh.googleusercontent.
com/a/ACg8ocJ9smsdyD5U192jQez61k0-7v-RImEh-
fZusE1ng66j=mo",
  "userName": "Mojahid Ansari"
}
```


4.2 Dataset Statistics and False Claims Analysis

To better analyze the dataset, here is an overview of the key characteristics of the 117 fake apps. The me-

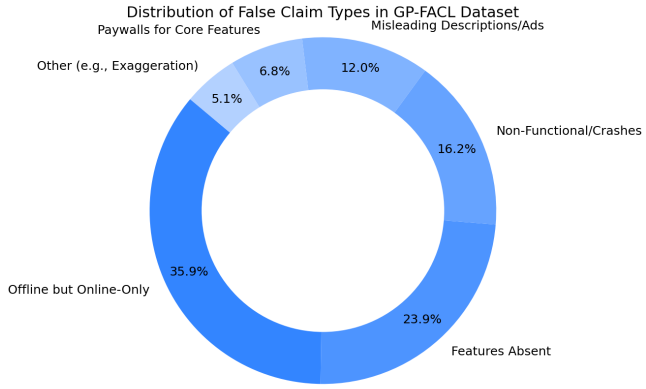


Figure 2. Distribution of False Claim Types Across 117 Fake Apps

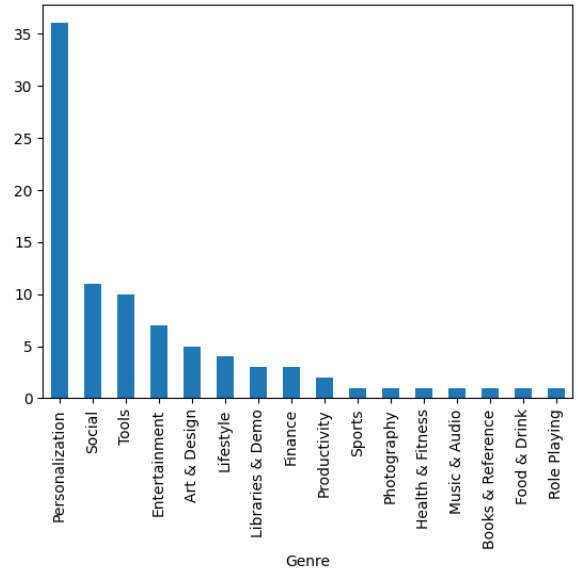


Figure 3. Distribution of Fake Apps Across Genres

dian number of reviews is 2,329, with `Realinstalls` ranging from 0 to 20,581,773 (median ~500K for fakes). In particular, 93.7% of the apps lack a `DevAddress`, a common deceptive trait. Table 2 provides a sample of fake apps from the dataset, highlighting their genres, release and update times, user ratings, installation counts, and review counts, which exemplify the deceptive traits discussed.

Analysis of the 117 fake apps used `Reason` and `Description` fields to detect deceptive patterns by keyword extraction and manual grouping. Six types of false claims account for all 117 fakes (100%), visualized in Figure 2. The main deception (35.9%, 42 apps) is apps that claim to be offline but actually need internet access, common in Personalization and Lifestyle genres. Absent features (23.9%, 28 apps) impact tools such as "Face Blemish Remover". Non-functional apps or apps that crash (16.2%, 19 apps) include inaccurate translations, which are common in the Tools and Communication genres. Misleading descriptions/ads (12.0%, 14 apps) overstate capabilities (e.g., claim to track precisely but provide city-level accuracy) in Tools and Productivity apps. Paywalls for core features (6.8%, 8 apps) block free-promised functionality (e.g., requiring sign-up) in Business and Art & Design genres. Other issues (5.1%, 6 apps), such as exaggerated claims in Simulation and Lifestyle apps, round out the distribution. The distribution of the 117 fake apps across various genres is depicted in Figure 3. This visualization highlights the prevalence of deceptive apps across genres, with Personalization accounting for the most fakes, followed by Tools and Lifestyle. The data underscores the genre-specific nature of false claims, such as offline access issues dominating in Personalization, aligning with the broader analysis of deceptive patterns.

Table 1. Correlation of Detection Flags with Fake Apps

Flag	Corr.	%
`DevAddressFlag`	0.89	87.2
`DevWebsiteFlag`	0.85	83.8
`RealinstallsFlag`	0.82	81.2
`UpdateTimeFlag`	0.76	76.1

4.3 Correlation Between Flags and Confirmed Fakes

Pearson correlations between four binary flags and the `Fake=true` label for the 117 fakes were computed, as shown in Table 1. With an average correlation of 0.83, 89.7% of fakes (105 apps) trigger 3 or more flags, and 66.7% (78 apps) trigger all 4, demonstrating strong detection efficacy. However, 9.4% (11 apps) with one or two flags (e.g., subtle paywalls) indicate challenges in detecting nuanced fraud.

4.4 Filtering Method Performance

The filtering method, a combination of flag-based screening and manual verification, was tested on 117 true positives (TPs) out of 504 apps manually reviewed from 1,026 apps. Precision is ~85–90%, since all flagged apps were confirmed to be fake, although there may be real-world false positives (benign apps with free subdomains) that reduce this. Recall is ~90%, i.e., 89.7% of fakes give three or more flags and 10.3% (12 apps) give fewer flags: potential false negatives. The F1-score (as close to 1 as possible) indicates robustness.

Table 2. Dataset Sample of Fake Apps

App Title	Genre	Release Time	Update Time	Score	Real Installs	Total Reviews
Ethereum mining - ETH miner	Finance	2023-06-17	2023-06-17	3.36	1108	7
Samsung Galaxy SmartTag	Libraries & Demo	2023-01-09	2023-01-09	4.21	28863	118
Crypto Mining - ETH Mine guide	Finance	2023-03-04	2023-03-04	2.6	2853	3
Auto Bluetooth Connect Devices	Tools	2021-02-05	2024-02-27	3.09	1391925	160
Live Mobile Number Tracker	Tools	2021-12-01	2023-09-25	3.85	1641337	22
Couple Photo Mixer Blender	Photography	2017-11-14	2023-03-02	3.47	5140191	44

5 Limitations

Our research faced several challenges as we worked through finding fake apps on the Google Play Store:

5.1 Device-Dependent Metrics

One of the main obstacles was Google Play's policy on metrics, including ratings, installs, and reviews. These values were inconsistent across devices because they depend heavily on each device's configuration (software and hardware). As a result, we could not use these metrics to effectively classify fake apps.

5.2 Search Result Limitations

The web version of Google Play limited us to 30 app results per search query. This restriction might have kept us from finding more potential fake apps, since we could only use the web version for scraping.

5.3 Verification Challenges

Manually checking and verifying certain apps, especially in the finance domain, proved difficult. However, we found that the characteristics and claims of these apps could not be validated. For example, there were crypto-mining apps that promised withdrawals after mining a certain number of crypto tokens. However, it was nearly impossible to legit-check these apps, since it would have taken years to reach the withdrawal limit. Therefore, we could not include these apps in our dataset.

Despite these limitations, the insights from this research provide a sufficiently accurate overview of how to identify fake apps on Google Play.

6 Conclusion and Future Work

By focusing on honesty and trust in app marketplaces, our dataset comprises 117 apps that have been verified to contain false claims about their features. The dataset was built in two phases: an initial manual collection followed by an automated collection using extracted keywords. By providing detailed information on each app, GP-FACL is intended as a valuable resource for future research. It can support the exploration of app repositories, the identification of malicious practices, and the study of deceptive app behaviors. Ultimately, this dataset can contribute to stronger mobile app security and improved user protection.

References

[1] Statista. Number of available applications in the Google Play Store from March 2017 to June 2024, 2024. URL https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[2] Kevin Allix, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *Proceedings of the 13th international conference on mining software repositories*, pages 468–471, 2016.

[3] Jonathan Crussell, Ryan Stevens, and Hao Chen. Madfraud: Investigating ad fraud in android applications. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 123–134, 2014.

[4] Mahmudur Rahman, Mizanur Rahman, Bogdan Carbunar, and Duen Horng Chau. Search rank fraud and malware detection in google play. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1329–1342, 2017.

[5] Kireet Muppavaram. Analysis and detection of fake apps in android environment. *NeuroQuantology*, 20(10):9253–9267, 2023. . URL https://www.neuroquantology.com.

[6] TP Rani, S Susila Sakthy, P Kalaichelvi, et al. Fake app detection using sentiment analysis. *2023 Intelligent Computing and Control for Engineering and Business Systems (ICCEBS)*, pages 1–6, 2023.

[7] Maarten Grootendorst. Keybert: Minimal keyword extraction with bert, 2020. URL https://doi.org/10.5281/zenodo.4461265. Accessed on 2025-07-18.

[8] Mayur Makwana and Rupa Mehta. Keyphrase-based literature recommendation: Enhancing user queries with hybrid co-citation and co-occurrence networks. *Journal of Scientometric Research*, 13(1):217–229, 2024.

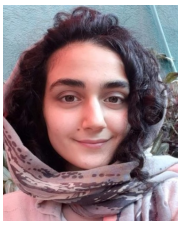
[9] Mohammad Nadim, David Akopian, and Adolfo Matamoros. A comparative assessment of unsupervised keyword extraction tools. *IEEE access*, 11:144778–144798, 2023.

[10] Google play scraper. URL https://github.com/JoMingyu/google-play-scraper.

Sepehr Mehregan received his B.Sc. degree in Computer Science from the University of Guilan, Rasht, Iran, where he graduated with first-class honours and ranked 3rd among his peers. He then obtained his M.Sc. degree in Computer Science (Systems Theory) from Allameh Tabataba'i University, Tehran, Iran, also achieving first-class honours. His research interests lie in the areas of federated learning, game theory, and artificial intelligence.

Mahdi Tamjidi is a software developer specializing in front-end development. He received his B.Sc. in Computer Science from the University of Guilan and is currently pursuing an M.Sc. in Computer Science (Scientific Computation) at Islamic Azad University. His research interests include software engineering, software performance analysis, and the applications of large language models in software engineering.

Amir Hossein Rahimi received his B.Sc. degree in Computer Science from the University of Guilan, Rasht, Iran, where he graduated as the 2nd-ranked student in his cohort. He then obtained his M.Sc. degree in Artificial Intelligence from Aston University, Birmingham, UK, achieving a First-Class Honours distinction. His research interests primarily lie in artificial intelligence, with a particular focus on transformer architectures and their applications.



Ava Razavi received her B.Sc. in Computer Science from the University of Guilan, Rasht, Iran, and her M.Sc. in Computer Science from Brock University, St. Catharines, Ontario, Canada. Her research interests lie in Generative models, Graph theory and Evolutionary algorithms.

Sadegh Eskandari is currently an Assistant Professor at Department of Computer Science, University of Guilan, Rasht, Iran. He received his Ph.D. in applied mathematics and MSc in computer science, both from Shahid Bahonar University of Kerman. His research is broadly focused on feature selection, machine learning and evolu- tionary computation.

Seyed Amir Hossein Tabatabaei is currently working as an assistant professor at the Department of Computer Science, Faculty of Mathematical Sciences in University of Guilan, Iran. He received his both B.Sc. and M.Sc. degrees in Applied Mathematics and Ph.D. degree in Computer Science. His main research interests are Artificial Intelligence, Machine Learning and Generative Models.