Obidroid: Monitoring the Android App Store for Unfair or Deceptive Practices

Luis Aguilar, Shreyas, Kristine Yoshihara, Marti A. Hearst
University of California Berkeley,
Berkeley CA 94720
{luis, shreyas, kristine_y, hearst}@ischool.berkeley.edu

1. INTRODUCTION

Google's Android platform is currently one of the most popular smartphone platforms in the world, with over 81% market share [1] and over 10 billion app downloads. Due to the platform's popularity, there are many potentially harmful applications that can steal user information, push malware, or otherwise cause injury to users. While Google's venue for apps, the Google Play Store, is constantly on the lookout for apps that are malware or viruses, there is currently no monitoring for apps that meet FTC requirements for unfair [2] or deceptive practices [3]. For example, the Brightest Flashlight Free Application was reprimanded by the FTC for deceptively continuing to share user data, even after users opted to keep their information private [4].

The FTC is tasked with protecting users against such apps, but this process has not been automated and they are reliant upon user complaints or media reports to identify potentially untrustworthy apps. This is due in large part to the volume of apps in comparison to the FTC's investigative bandwidth.

In **Obidroid**, we have built a predictive model based on app attributes that identifies or marks applications as **flagged** that may be engaging in unfair or deceptive practices. The system does not pass final judgment on the legality of an app, but rather is intended as a helpful assistant for initially flagging such apps, which can then be investigated more closely by a human expert.

We aimed to cast a wide net while flagging since the system is intended to augment, not replace human intervention: high recall is to be preferred over high precision. Hence, incorrectly flagging good apps is more acceptable than overlooking apps that should have been flagged.

The tool should be adaptable to reflect the constantly changing nature of user reviews and the underlying applications, and it should be able to be run on a periodic basis.

2. RELATED WORK

The Google Play Store currently screens traditional malware through an application called Bouncer, which periodically analyzes apps through dynamic analysis in a sandbox. Detection of unfair and deceptive apps is largely left to crowdsourcing and users are expected to evaluate the privacy policy and requested permissions before installing an app [5]. Given sufficient severity and volume of complaints, the FTC evaluates apps for unfairness and deception, according to the following criteria: 1. Unfairness: Substantial injury to consumers that cannot be reasonably avoided and is not outweighed by countervailing benefits and 2.) Deception: material harm to the consumer caused by misrepresentation, omissions, or practices likely to mislead. This evaluation is typically performed through a mix of qualitative and technical evaluation of the discrepancy between what the app should be doing and its actual actions. [2][3][4].

Much of the recent research on privacy and untrustworthy apps has focused on the relationship between permission, malware, and user expectations [6]. Previously, Kuehnhausen and Frost developed a system to detect malicious apps based on ratings, permissions, and analysis of review spelling and sentiment [7]. None of these studies specifically use FTC criteria, focusing instead on indicators of traditional malware.

3. FEATURES AND DATA

We obtained data by scraping app profiles from the Google Play store, which were then hand labeled according to FTC criteria listed in the previous section, and scaled across each feature.

Currently, the FTC analyzes apps based on Google Play Store attributes to search for apps that are indulging in unfair or deceptive practices. We assessed the validity of those attributes as features for flagging the apps and then created additional features to improve the predictive power of our model.

The features shown in Table 1 were useful for building a statistical model. We evaluated the performance of both parametric and nonparametric methods using these features, including K Nearest Neighbor, Gaussian Naïve Bayes, decision tree based ensemble methods, and Support Vector Machines. K Nearest Neighbor weighted by distance achieved the highest average adjusted accuracy, based on 13 trials, in which the classifier was trained on set of 36 apps and tested against a set of 10 apps, both sets having a 1:1 fair to unfair ratio. The test set is currently small because only limited data was available.

Table 1. Features extracted from written reviews of apps.

Feature	Description	Intuition / Origin
revSent	Aggregate review sentiment, by classifying each sentence of a review into positive, negative and neutral[8]	NLP inspired
revLength	Length of review (character count)	NLP inspired
avgRating	Average rating of the app	-
hasPrivacy	Whether the app has a privacy policy or not	FTC inspired
Has Developer Website	App has an associated developer website	FTC inspired
Count Multiple Apps	App creator has multiple apps on the app store	Reliability of the app creator
Installs	Total installs of each app	-
Count Exclamation	Count of Exclamation Points	NLP inspired

countCapital	Count capitalized words in a review	NLP inspired
countAdjective	Count of Adjectives in a review	NLP inspired
Count Negative Words	Count the number of negative words from a curated list	NLP inspired
unigrams	Presence of curated *malindicator* words	NLP inspired
bigrams	Top 20 bigrams via likelihood ratio measure	NLP inspired
trigrams	Top trigrams based on raw frequency	NLP inspired

4. FINDINGS & RESULTS

In evaluating the performance of our model, we preferred models with lower false negatives over false positives, in order to cast a wider net. Based on this metric, the average prediction accuracy for the best model (K Nearest Neighbor) on the entire dataset was 90% using 4 fold cross validation. The review sentiment alone gave a model with 86.25% accuracy, but adding other features increased the performance of the model.

Our model revealed that the feature *installs* exhibits a behavior (as shown in Table 2) such that values lower than 3M are good predictors for flagged applications and higher values predict unflagged applications. In contrast, other features were good indicators for either one class or the other. Perhaps not surprisingly, negative review sentiment, many words in all caps, and long reviews were good predictors of flagged apps. Counterintuitively, high average ratings were observed on both flagged & unflagged apps, indicating that average rating, although sometimes trusted by users, is not a good predictor. We further clustered the apps using Multidimensional Clustering (MDS) in order to determine where our model was failing to classify the apps in the right category.

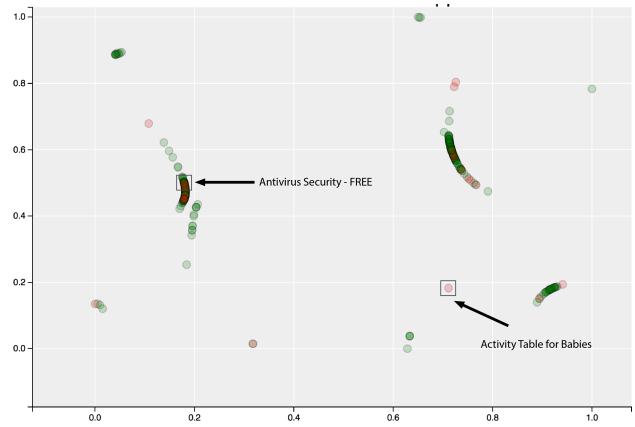

Figure 1. MDS clustering of all apps in our dataset. Red dots are flagged apps.

MDS clustering revealed that certain flagged apps, such as the "AntiVirus Security – Free" app, were clustered closely to apps that were not flagged. These apps may be more prone to be mislabeled. Adding or refining features may increase the distance between these flagged apps and apps that were not flagged.

We generated a list of the most informative features from the Obidroid Model to evaluate the features extracted from the app's attributes. Table 2 contains a summary of performance of those key features.

Table 2. Most Informative Features

Feature	Feature Value	Feature Performance (flagged/unflagged ratio)
Installs	Installs=3,000	9:1
	Installs=30,000	6:1
	Installs=3,000,000	1:2
revSent	revSent = -17	8:1
	revSent=-10	2:1
countCapital	countCapital=9	3:1
revLength	revLength=800+	2:1

5. FUTURE WORK

Based off of these results, we can expect the Obidroid model, on average, to flag 9 out of 10 potentially untrustworthy apps. However, these results are based on a very small training set and test set and so may only be suggestive. A tool based on this model could be run to periodically scale down the task of monitoring the exponentially increasing number of apps on the App Store.

6. ACKNOWLEDGMENTS

Our thanks to Deirdre Mulligan, Doug Tygar, Jen King, Serge Egelman, and Morgan Wallace for their support in this project.

7. REFERENCES

[1] Dredge S. 2013. Android takes record smartphone share at expense of iPhone and BlackBerry. Retrieved May 1, 2014 from at guardian.com/technology/2013/oct/31/android-record-smartphone-share-iphone-blackberry.

[2] FTC. 1980. FTC Policy Statement on Unfairness. Retrieved May 1, 2014 from ftc.gov/ftc-policy-statement-on-unfairness.

[3] FTC. 1984. FTC Policy Statement on Deception. Retrieved May 1, 2014 from ftc.gov/ftc-policy-statement-on-deception.

[4] FTC. 2014. In the Matter of Goldenshores Technologies, LLC, and Erik M. Geidl. Retrieved May 1, 2014 from ftc.gov/enforcement/cases-proceedings/132-3087/goldenshores-technologies-llc-erik-m-geidl-matter.

[5] Cuadrado F. and Duenas, J.C. "Mobile application stores: success factors, existing approaches, and future developments." *Communications Magazine, IEEE* 50(11), 2012.

[6] Felt A., et al. Android Permissions Demystified. Proceedings of the 18th ACM Conference on Computer and Communications Security, 2011.

[7] Kuehnhausen M and Frost, Victor. Trusting smartphone Apps? To install or not to install, that is the question. CogSIMA, IEEE International Multi-Disciplinary Conference, 2013.

[8] Hu M. and Liu B. "Mining and summarizing customer reviews. "Proceedings of the Tenth ACM SIGKDD, 2004