Identifying Spam in the iOS App Store

Rishi Chandy
Carnegie Mellon University
rishic@cs.cmu.edu

Haijie Gu
Carnegie Mellon University
haijieg@cs.cmu.edu

ABSTRACT

Popular apps on the Apple iOS App Store can generate millions of dollars in profit and collect valuable personal user information. Fraudulent reviews could deceive users into downloading potentially harmful spam apps or unfairly ignoring apps that are victims of review spam. Thus, automatically identifying spam in the App Store is an important problem. This paper aims to introduce and characterize novel datasets acquired through crawling the iOS App Store, compare a baseline Decision Tree model with a novel Latent Class graphical model for classification of app spam, and analyze preliminary results for clustering reviews.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Experimentation, Measurement

Keywords

review spam, opinion spam, mobile apps, fraud detection

1. INTRODUCTION

Launched in 2008, the iOS App Store now lists over 500,000 apps for Apple's iPhone and iPad mobile devices [1]. The App Store contains both free and paid apps, with Apple taking 30% of revenue from app purchases. Since Apple's mobile devices only support apps downloaded from this store, application developers have access to a large audience of potential customers. Indeed, a popular app, such as "Angry Birds," can generate millions of dollars.

Developers of spam apps (malicious developers) are primarily interested in gaining monetary profit or leaching valuable user data, such as address book contacts. Popular, seemingly legitimate apps can leak user data quietly [2, 4], so it is feasible that spam apps would attempt to do the same.

Figure 1: App and developer features from the Labeled E&L dataset used for learning the baseline Decision Tree

App Features
App ID
Developer ID
Price
Category ID
Release Date
Current Version

Developer Features
Developer ID
Number of Apps
Avg App Rating
Avg Number of App Versions
Avg Review Helpfulness
Proportion of Free Apps

In the App Store, each app has its own webpage, which displays app price, screenshots, description, ratings and text reviews left by users who downloaded the app, and related metadata. Ratings are integer "stars" in the range 1-5. Similar to other online shopping platforms, positive reviews are crucial for convincing potential customers to purchase the app. Fraudulent reviews could deceive users into downloading potentially harmful spam apps or unfairly ignoring apps that are victims of review spam.

A malicious developer could post spam reviews by using several throwaway iTunes user accounts i.e. "sockpuppets". Apple has attempted to decrease the frequency of spam by requiring users to purchase and download an app before being able to review it. However, sockpuppet user accounts can still be created using iTunes Gift Cards, and the potential for profit and stolen user data could justify the cost.

For malicious developers, spamming the App Store can be beneficial and is not difficult, so automatically identifying spam in the App Store is an important problem. The goals of this paper are to classify app spam in a supervised setting with limited labeled data, and to cluster reviews in an unsupervised setting. Our main contributions are the introduction and characterization of novel iOS App Store datasets, comparison of a Decision Tree model and novel Latent Class graphical model for classification of app spam, and preliminary results on clustering reviews with analysis.

1.1 Related Work

One of the first modern data mining approaches to detecting opinion spam is presented in [6]. They studied an Amazon review data set, and categorized opinion spam into three types: untruthful reviews, brand-only reviews, and

Node	Features	CPD
f_u	user_avg_rating, user_num_rev	conditional Gaussian
f_a	app_avg_rating, app_num_rev	conditional Gaussian
f_r	$I(stars <= 2), I(star = 3), I(stars >= 4)$	NA
f_d	dev_num_app, deb_avg_rating,	conditional Gaussian
I_a, I_d, I_u, I_r	binary class indicator	CPT

Table 2: Features and CPD

non-reviews. Because brand-only reviews and non-reviews are easily recognizable by humans, the problem of detecting reviews of those types can be transformed into a supervised classification problem. However, untruthful reviews are much harder to distinguish by inspection and hence manually creating labels becomes infeasible. Their approach used duplicate content as the main indicator for detecting untruthful reviews, since they found that untruthful reviews are likely to be reposted verbatim repeatedly.

Subsequent efforts focused on feature engineering and natural language processing. Researchers used text features to examine reviews in domains such as movies and products. A classifier for deceptive opinion spam in TripAdvisor was developed in [9], emphasizing psycholinguistic methods and text analysis. They also confirmed that deceptive opinion spam is difficult for humans to identify reliably.

Table 1: Sizes of "Top Apps" (TA), "Entertainment & Lifestyle" (E&L), and "Labeled E&L" datasets.

	TA	E&L	Labeled E&L
# Apps	691	2,399	114
# Reviews	6,282,626	37,035	33,134
# Users	4,416,823	36,705	32,932
# Developers	412	2,002	114

Instead of finding item or review spam, [7] attempted to identify the reviewers who spread spam by looking at suspicious rating behavior. Their approach was based on heuristic models of review patterns. In addition, they also found that spam can significantly affect product ratings.

The techniques based on a review graph developed in [10] sought to avoid text features entirely, while accounting for the interaction among reviews, reviewers, and the review subjects (online stores) in a ResellerRatings dataset. Their iterative algorithm computed scores for their definitions of reviewer trustiness, store reliability, and review honesty.

A benign technique to increase app downloads was explored in [5], which found that Sunday evening is the best time to release a new game due to app usage patterns. The validator implemented in [4] tried to identify a subset of spam apps, those that leak private information, by analyzing the app executable.

2. DATASETS

The datasets consist of all reviews for selected apps crawled from the Apple iOS App Store in 2012. From the review data, we computed metadata for apps, developers, and users who post reviews. We obtained two datasets: the "Top Apps" (TA) dataset containing reviews and metadata for the most popular iPhone and iPad apps according to Apple's leaderboards, and the "Entertainment & Lifestyle" (E&L) dataset containing reviews and metadata for all iPhone and iPad apps in the Entertainment and Lifestyle categories. In addition, we created a third dataset, "Labeled E&L", which contains a randomly chosen subset of apps from "E&L" having more than twenty reviews with app spam binary labels acquired through manual inspection. The size of each dataset is shown in Table 1.

Figure 2 shows CCDFs for the TA and E&L datasets. In Figure 2a, we observe that the CCDFs go flat at integer rating values, with noticeable drops immediately before these values. As expected, the TA CCDF remains higher than the E&L CCDF at all rating levels in this plot, with the difference increasing with rating. This is because the TA dataset contains many of the "best" apps in the entire store, while the E&L dataset simply contains all apps from two categories. Also, the drop near 4.5 in the TA CCDF (circled in red) indicates that many users would find an intermediate rating between 4 and 5 useful for delineating the very best apps. Figure 2b confirms this near the 4.5 rating level and also shows how the "best" apps differ from the rest.

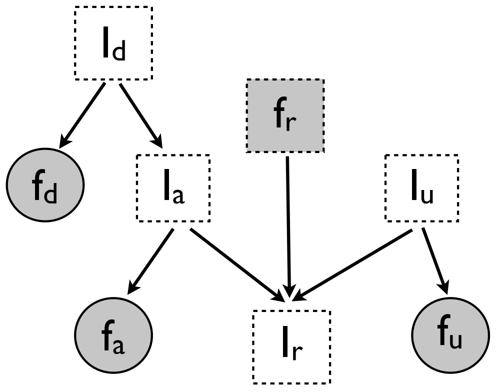


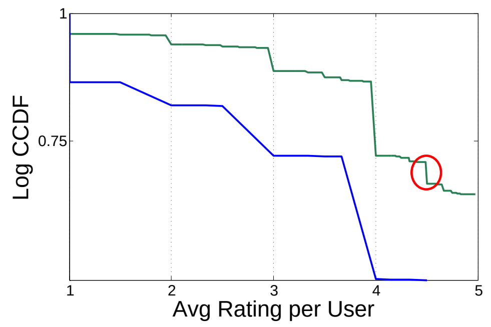
Figure 3: Latent class model for review, user, app, and developer. Squares represent discrete variables and circles represent continuous variables. Shaded nodes are observed.

3. METHODS & RESULTS

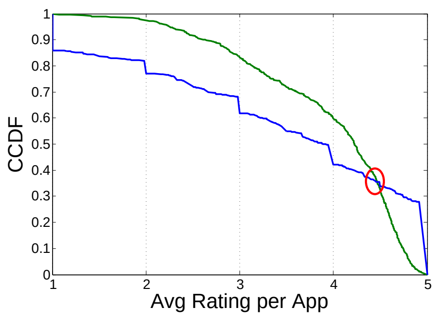
As a baseline method to classify app spam, a pruned Decision Tree was trained on app and developer features from the Labeled E&L dataset shown in Figure 1.

In addition to the decision tree, we propose a simple latent

Figure 2: CCDFs from the E&L (blue) and TA (green) datasets. Particularly interesting regions are circled in red.



(a) CCDF for users' average rating



(b) CCDF for apps' average rating

class model to capture the relationship among the review, app, user, and developer. Figure 3 shows the structure of the graphical model. For each entity of interest, we assign it a feature node and a latent node which represents the latent class. The latent class model assumes that the feature is generated from the unobserved class, and is independent of other nodes given the class. Furthermore, we assume that the developer only directly affects the app. All latent class variables in this model are chosen to be binary. I_a indicates good or bad apps, I_d indicates good or bad developers, I_u indicates normal or malicious users, and I_r indicates truthful or spam reviews. Table 2 summarizes the features and the conditional probability model at each node.

For simplicity and interpretability, we choose the two most common used features for each observed node. We choose Linear Gaussian to model the conditional probability of a feature node given its latent class. We also simplify the review feature to be a class indicator of high, middle and low. Hence, it is convenient for us to put priors on the CPT based on heuristics such as if conditioning on a dishonest user, a high quality app, and a low review, the review is more likely to be spam.

	$P(I_u)$	$P(I_d)$	$P(I_a\|I_d=0)$	$P(I_a\|I_d=1)$
0	0.12	0.13	0.91	0.16
1	0.88	0.87	0.09	0.84

Figure 4: Learned parameters on latent nodes. The first column contains the value of the variable.

3.1 Supervised Results

In the supervised setting with labels on apps, we treat I_a as observed during training, and as unobserved during classification on the test set. The Labeled E&L dataset was split randomly 50/50 between the training and test data. The baseline Decision Tree achieves 41.4% classification error on testing, with false positive rate 35.7% and false negative rate 46.7%. On this dataset our Latent Class model achieves 26.4% classification error on testing with false positive rate 6.3% and false negative rate 40.9%.

3.2 Unsupervised Results

The unsupervised learning is run on the E&L dataset. The goal is to cluster reviews using the latent node I_r. We

I_d	$\mu(f_d)$	$\sigma^2(f_d)$
0	(2.5, 1.5)	(0.6, 0.7)
1	(4.2, 1)	(0.01, 0.18)

(a) Parameters of $f_d|I_d$

I_a	$\mu(f_a)$	$\sigma^2(f_a)$
0	(3.3, 71)	(1.2, 6701)
1	(4.2, 432)	(0.01, 90500)

(b) Parameters of $f_a|I_a$

I_u	$\mu(f_u)$	$\sigma^2(f_u)$
0	(3.9, 1)	(0.01, 2.13)
1	(4.0, 1.5)	(0.51, 1.47)

(c) Parameters of $f_u|I_u$

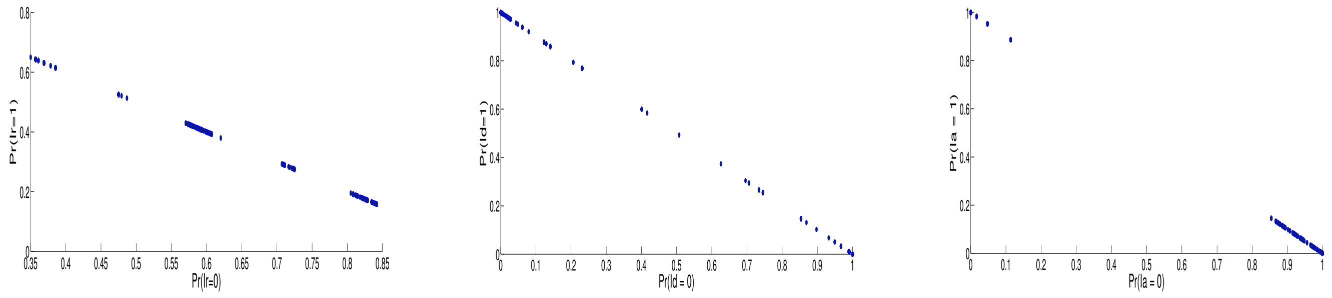
Figure 5: Learned parameters on feature nodes. First column is the value of parent node. Figure 5a shows the parameters of the (*avg_rating*, *num_reviews*) pair, while Figures 5b and 5c show the parameters of the (*avg_rating*, *num_apps*) pair. We restrict the covariance matrix to be diagonal during the learning.

start with a uniform prior for I_a, I_d, I_u. We set a prior on $P(I_r|I_a, I_u, f_r)$ to encode common beliefs on a review's truthfulness based on user's honesty, review's rating, and app's quality. We run Expectation Maximization [3] for 6 iterations with a Junction Tree inference algorithm provided by the Bayesian Network Toolbox (BNT) [8]. Note that, although our goal is to cluster spam reviews, having other latent nodes in the model provides a clustering on the users, apps, and developers as a free byproduct.

Figures 4 and 5 show the parameters learned from EM. In Figures 5a and 5b, the conditional mean of *average_rating* of apps and developers agrees with the intuition that higher quality apps and developers receive higher ratings. Apps from class 1 (good quality) receive more reviews than apps from class 0, and the variance is much higher in class 1. However, we notice that the *number_of_apps* feature for developer is similar for both classes, because most of the developers have only 1 or 2 apps in this dataset. Also, the parameters of user features are similar for both classes because most of the users only posted one review. Therefore average rating of users does not provide enough information for clustering users. The marginal probability of latent class in Figure 4 shows that the prior belief on users, apps, and

Figure 6: Plots for marginal probability of latent nodes I_r, I_d, I_a. Each point corresponds to a sample record, and the axises are the marginal probability of the corresponding latent class.

(a) Marginal of I_r (b) Marginal of I_d (c) Marginal of I_a



developers is heavily favored toward 1 (good class).

Figure 6 shows the projection of the data onto a 2D space using the marginal probability of the latent variables I_r, I_d and I_a. In Figure 6a, reviews are well separated into 5 groups. One big cluster is centered around 50% which explains the ambiguous nature of some reviews. In Figure 6b, most of the developers fall into two clusters on both ends, and some developers are scattered in between. Figure 6c shows that apps are strongly clustered on two ends. On node I_u (not shown), however, most of the users fall into the same cluster. Because almost all users in this dataset have only one review, the user features by themselves are not very effective for clustering users.

4. CONCLUSIONS

In this paper, we characterize novel iOS App Store datasets, drawing observations from average rating CCDFs for apps and users. We propose a latent class model with interpretable structure and low complexity. On the labeled data set, even though we use the simple Linear Gaussian parameterization, it still achieves significantly higher accuracy than a baseline Decision Tree. On the unlabeled data set, it succeeds in clustering the apps and reviews into well separated groups. Future work could explore extending our Latent Class graphical model to adopt more features with a different parametrization.

5. ACKNOWLEDGMENTS

This work is partially supported by the Information Network Academic Research Center (INARC), part of the Network Science Collaborative Technology Alliance (NS-CTA). The authors would also like to thank Professors Christos Faloutsos and Geoffrey Gordon for their mentorship and support.

6. REFERENCES

[1] Apple. Apple iPhone 4S Announcement. `http://events.apple.com.edgesuite.net/11piuhbvdlbkvoih10/event/index.html`, 2011.

[2] Nick Bilton. Disruptions: So Many Apologies, So Much Data Mining. `http://bits.blogs.nytimes.com/2012/02/12/disruptions-so-many-apologies-so-much-data-mining`, 2012.

[3] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[4] Peter Gilbert, Byung-Gon Chun, Landon P Cox, and Jaeyeon Jung. Vision: automated security validation of mobile apps at app markets. In *Proceedings of the second international workshop on Mobile cloud computing and services - MCS '11*, page 21, New York, New York, USA, 2011. ACM Press.

[5] Niels Henze and Susanne Boll. Release your app on Sunday eve: finding the best time to deploy apps. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*, page 581, New York, New York, USA, 2011. ACM Press.

[6] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the international conference on Web search and web data mining - WSDM '08*, page 219, New York, New York, USA, 2008. ACM Press.

[7] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10*, page 939, New York, New York, USA, 2010. ACM Press.

[8] Kevin Murphy. The Bayes Net Toolbox for MATLAB. *Computing Science and Statistics*, 33, 2001.

[9] M. Ott, Y. Choi, Claire Cardie, and J.T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 309–319. Association for Computational Linguistics, 2011.

[10] Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. Review Graph Based Online Store Review Spammer Detection. In *2011 IEEE 11th International Conference on Data Mining*, pages 1242–1247. IEEE, December 2011.