UNDERSTANDING TECHNOLOGICAL ABUSE: AN EXPLORATION OF CREEPWARE

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

MSc.

by

Paula Barmaimon Mendelberg and Daniel Nissani

May 2020

ABSTRACT

Prior research has explored Intimate Partner Surveillance (IPS) apps, and developed a machine learning model to detect these apps on victim's phones. Our exploration bolsters these detection systems by providing insights on a dataset of millions of devices and their installed apps. Moreover, we provide descriptions and patterns of the "creepware" ecosystem that is currently present in the app marketplace.

With a taxonomy of the creepware space, we attempt to build a machine learning model to label the over three million apps that exist in our dataset. This would have helped in the development of a feature space to classify devices based on their theoretical use cases. However, due to a lack of data and issues with data governance, we instead present an analysis of the previously made taxonomy using LDA, as well as an attempt to cluster devices using a feature space created out of the current existing taxonomy.

0.1 Introduction

Prior work by the Intimate Partner Violence (IPV) group at Cornell Tech has qualitatively validated that technological methods of surveillance were being implemented in these settings.[2] Furthermore, the group, specifically in Chatterjee et. al. [1], implemented what they called a query snowball, which involved seeding searches in an app-store with queries such as, "catch my cheating girlfriend." Through this query snowball, the group manually identified applications that were used in IPS settings, with verification from forum and blog post conversations. By hand labeling the apps, the team built a logistic regression model to act as a scanner for IPS applications on mobile phones. By seeding such queries, suggestions of additional queries were given by Google, obtaining with this snowball method, a wide range of relevant queries. The apps that were ranked when searching with each of these queries were manually checked by the group, identifying some as IPV related apps.

Previous research by Roundy et. al., explored the "creepware" ecosystem. Norton Security Group provided both an anonymized dataset and an algorithm to search the dataset. The dataset has 27.7 million devices with 10.8 million unique apps that were installed 4 billion times. Each line of the dataset represents app installations on an anonymized device, which contains a string of tuples formatted as (app-id, timestamp), where timestamp represents when a phone was scanned by the vendor's security application.[4]

The algorithm created by the vendor was seeded with 18 applications that the IPV group at Cornell Tech deemed unambiguously intended for IPS use.[1] This algorithm scored applications based on their co-occurrence with the seed

set of applications, and we explored the top 1000 of these to develop a codebook and test hypotheses we had about the dataset.

With the taxonomy developed in Roundy et. al., we attempted to create a feature set for the applications that had been coded in order to classify apps. This resulted in a set of 1091 featured apps, and a set of 80,476 apps that would be labeled by our supervised machine learning models. These efforts did not purport much success, thus we evaluated the taxonomy using latent dirichlet allocation through gensim.[5]

With this same taxonomy, we attempted to create a feature set for the devices, where a feature would be a category or subcategory in the taxonomy, and an instance would include the count of apps included in a device profile for each category or subcategory. 153,884 devices where used, with 4,085 labelled apps classified in 10 categories and 56 subcategories. These featured devices were attemped to be clustered using k-means, in the hopes of identifying clear profiles (victim vs. abuser) or identifying categories or subcategories of apps that were appearing with high correlation both with clear defense apps or clear abuse apps, indicating the ambiguity of some of the categories or subcategories and the possible need of further study into understanding the nature of these use cases.

0.2 Related Work

Intimate Partner Violence (IPV) is a prevalent global problem that affects millions of people worldwide, with recent reports suggesting that one in three

women and one in six men experience IPV at some point in their lives.[2] Previous research by the IPV team at Cornell Tech consider how abusers use technology to perpetuate their abuse.

In Freed et. al[3] the team conducted interviews with 40 IPV professionals and nine focus groups with 32 survivors of IPV who have visited Family Justice Centers to receive legal services, social services, and support. This paper presents an analysis of the multifaceted role that technology plays in the IPV ecosystem. They show a complex set of socio-technical challenges that emerge from the intimate nature of the relationships involved and the physical and digital complexities associated with managing shared families and social circles.

In Freed et al.[2], the teams analysis shows that common threat models do not anticipate attackers who possess intimate knowledge of, and access to, victims, as they do in intimate partner violence cases. In these scenarios, abusers are often legal owners of the victims devices or online accounts and take advantage of this fact. Abusers also compromise victims accounts by guessing their passwords or forcing them to disclose them, which in turn enables digital tracking, installation of spyware, denial of access to devices or the Internet, and more. The paper also explains how abusers intimidate and harass their victims through hurtful messaging or posts, or publicly reveal sensitive information in order to humiliate and harm them.

Both these papers inspired the need to understand the ecosystem of applications that abusers are using. As stated in Freed et. al.[2], the technology used by abusers is usually simple. Thus, if we are able to understand the ecosystem of apps from easily accessible app marketplaces, such as Google PlayStore, then we will get a better sense of future applications that will inevitably come out,

including those remarketed as older apps. Although not spoken about in depth in this paper, our team has reason to believe that the ecosystem is evolving as regulations become more strict. Thus, creating a taxonomy for the types of applications is an imperative.

Chatterjee et. al.[1] is an in depth study of the IPS spyware ecosystem. Using web crawling and manual labelling, the team managed to find and label apps that are potentially dangerous in IPS settings, ultimately identifying several hundreds of such IPS-relevant apps. While there are many overt spyware tools, most are "dual-use" apps — they have a legitimate purpose (e.g., child safety or anti-theft), but can be easily used for spying purposes. Additionally, the team manages to document a considerable amount of online resources available to educate abusers about exploiting apps for Intimate Partner Surveillance. It also showed how some dual-use apps encourage their use in IPS via advertisements, blogs, and customer support services. Existing anti-virus and anti-spyware tools are analyzed, most of them failing to identify dual-use apps as a threat.

Moreover, Chatterjee et. al.[1] contributes a logistic regression model that can scan phones and detect IPS applications. This is something many vendors have tried to do, but have been less successful. This supervised machine learning model was made possible by manually coding many apps. Thus, our research hopes to help the team move in the direction of having apps automatically coded with the help of the vendor's algorithm.

0.3 Methods

We used a mix of qualitative, quantitative, and technological methods for this research. Not only are the methods described here research methods, such as the qualitative coding, but also real technological methods, such as Search Engine Optimization (SEO) and machine learning.

0.3.1 Developing Ground Truth

The vendor provided an algorithm that ranks apps based on how associated they are with a seed set of apps. The initial seed set was provided by the IPV team at Cornell Tech. It was a set of 18 apps that were manually classified as particularly malicious for Intimate Partner Surveillance (IPS) settings. One question is whether we can add in new apps to the seed set, which apps those should be, and how the algorithm will perform with a new seed set.

After one run of the algorithm, the vendor provided 51 apps that were hypothesized as quite malicious from a larger set of the top 1000 apps. Each of the 51 apps were manually checked by exploring the app's apk data, videos associated with the app (if available) and the app's description. At times, reviews and other consumer information were used as well.

0.3.2 Codebook

With the purpose of understanding the creepware space, the team coded the top 1000 apps after one run on data provided by the vendor. Four researchers

coded the 1000 apps. In the first round, coders coded a shared batch of apps and met later to decide on appropriate code categories. The group did the same for an additional three rounds (the last round consisting of 35 apps) making 110 the total list of apps coded by the whole team. In each round the team refined the codebook with new categories or subcategories for exceptions found in the round that appeared to be common cases. After each round, one researcher tested the inter-coder reliability and found that the team performed at near perfect levels. After the 110 apps were coded, the team went and coded all of the remaining apps. Apps were given only one code.

0.3.3 Analyzing Timestamps

All of the apps on the devices, provided by the vendor, were from the Google PlayStore. Each timestamp for an app was relative to the installation time of the security app provided by the vendor. The relative-time is set to zero seconds for all apps installed prior to the vendor's security app. All other apps have a timestamp, in seconds, relative to the security app's download time. This scan happened in three ways, either a batch scan done daily, by the user initializing a scan, or, if the vendor's app is on in the background, everytime an app is installed. Thus, one of the questions we want to explore is whether these time stamps can be used as a proxy for installtion times or not. To analyze the timestamps of these apps, and evaluate whether they were a good proxy for installation time, we primarily used visualization techniques such as histograms and time series charts.

0.3.4 Utilizing Search Engine Optimization (SEO)

The method used in previous studies by the team to gather new apps in the IPV field consisted of a snowball search of terms on Google.[1] Although this is a good method for discovering new related search terms, it is not necessarily exhaustive, since Google's algorithm will suggest terms close in intent to the term searched for in the first place, usually longer tail terms in the same family of keywords. This means, that if we searched for the term 'how to check cheating girlfriend' suggested terms could include 'how to catch your girlfriend cheating on facebook' or ' check cheating girlfriend app'. Although these terms will show different results, they all have the same intent: looking for ways to check whether your girlfriend is cheating, and so the group of keywords all have a similar intent according to the Google algorithm. In suggested terms, we will never obtain families of keywords like 'how to hack my girlfriends phone', or 'check where my girlfriend is', since the intent is far (although all part of IPV cases) and therefore will not be suggested as related terms. In order to have a full mapping of all related keywords and keyword families, a thorough keyword research needs to be done, using tools like Google Keyword Planner or SemRush.

0.3.5 Co-Occurrences

In order to understand trends of co-occurring apps, we developed two heatmaps. One heatmap showed a 1000 x 1000 matrix of the top 1000 apps, showing a stronger color in the heatmap scale for higher count of devices where both apps appeared to be installed. In this example, and in order to get a better color repre-

sentation of the heatmap, the diagonal was not taken into consideration, that is, the total count of installs for a certain app, (or in the matrix, an app co-occurring with itself). The second heatmap was a 48 x 48 matrix of the low level categories the apps were coded as. The stronger the color in the heatmap scale, the more simultaneously installed apps from both of these categories. To normalize the co-occurrences, the vendor decided to use Point-Wise Mutual Information (PMI), a technique that measures how much information both co-occurring apps share about one another probabilistically. Point-Wise Mutual Information looks at the joint probability of both apps occurring on a device and then normalizes using the prior probabilities of either app appearing on a device independently. Further normalization is done by taking the logarithm, as shown below:

$$PMI(x;y) = \log(\frac{p(x,y)}{p(x)p(y)})$$

Thus, when the two variables are independent, we get a value of 0.

0.3.6 Data Collection & Classification

After building our taxonomy and having labeled 4000 applications, we wanted to find an automated way of collecting data on applications to build a feature set. This feature set would allow us to build a supervised machine learning algorithm to label the over 3,000,000 applications we have in our dataset.

To collect this data, we built a webscraper using the BeautifulSoup library on Python that collected data on apps that were available on APKPure, an app store that even contains apps that have been deleted from the Google Play store.Since

the indexed URLs for these apps where not always following a pattern, they were difficult to guess and scrape directly from Google. Scraping the app store in APKPure directly was also unsuccesful, since these apps where not appearing in APKPure's search results, probably due to previous efforts by the team to remove these apps from app markets. Due to these complications, we instead used data found from previous research. Although not comprehensive, it acted as a proof of concept for further research to be done.

For the labeled apps, we only used high level categories from the taxonomy described in Roundy et. al.[4] Then we grabbed market place data from a previously used and maintained dataset. This dataset was generated and used in previous research by the Cornell Tech IPV team in Chatterjee et. al.[1] For the 80,467 unlabeled apps, Norton provided the title and description for each app.

Using this data, we concatenated title, description, and, when available, comments and genre. In algorithm development, the labeled dataset was split into 75% train and 25% test. Using sklearn's TFIDFVectorizer, we generated a corpus of words from the training data. Stopwords and words occuring at most 5 times in all the apps were removed. All words were lowercased, as well.

Using 5-fold cross validation measuring against precision, recall, and f1-score, we assessed the viability of 4 different algorithms from sklearn: Multionmial Naive Bayes, Logistic Regression, Random Forest, and SGDClassifier. Once an algorithm was chosen, we performed RandomSearch from sklearn to choose the best hyperparameters for the model. This final model was then trained on the entire labeled dataset, thus increasing the corpus size for the TFIDFVectorizer.

To validate the final model's performance, the unlabeled apps were labeled by the algorithm and a balanced random sample of 100 datapoints, 10 for each class, was taken from the labeled applications and manually labeled by two researchers indenpendently to validate the results. After performing model validation, we also did an analysis as to whether the labels accurately clustered the data. Thus, we employed LDA from gensim for the topic modeling task.

0.3.7 Data Collection & Clustering

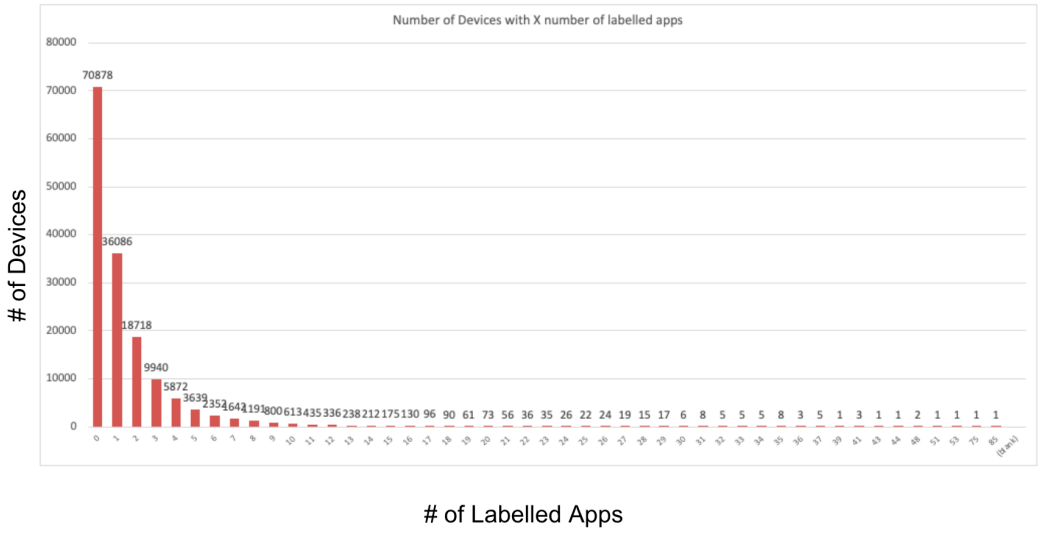

Figure 1: Distribution of devices based on the number of labelled apps on a device.

In order to cluster devices, the dataset with one device per row and installed appids for each device was used. For each device, a count of labelled installed apps for each category was calculated, having the categories be the features in the dataset. The inital dataset however, included 153,884 devices (all with at least 10 installed apps), which ranged in having anywhere from 0 labelled apps to 85 labelled apps. Most devices had 0 labelled apps (70,878), followed by 1

labelled app (36,086 devices). Since 69.5% of the devices had 0 or 1 labelled app, we can already see how difficult it would be to cluster devices with such limited data.

In addition to this, the labelled dataset used was that of 4,085 apps, which is heavily skewed towards 'none' apps (1847 of them) and 'surveillance' apps (1000) followed by 'info-extration' apps (297). If we take into consideration that most devices have bearly 1 or 2 labelled apps (if any), and that it is very likely that these apps are either 'none' or 'surveillance', the clustering will probably follow this distribution and cluster around those categories. Although we were convinced that a fully functioning classifier was necessary to fully describe the devices and therefore attempt any kind of device profiling, we went on to try and cluster devices with the current available data and either find proof of the former, or indeed find some indication that device profiling (abuser, victim, kid) is possible in this way.

If this clustering were to have worked, we could have also explored categories co-appearing in same clusters, and questioned whether we can really identify a specific kind of user (victim or abuser) for each category. While some categories clearly describe victim type of apps (defense apps, for example), and others clearly describe abuser type of apps (harassment apps, for example), others are not so easily idetifiable. For example, control apps, that can hide icons of other apps installed on the phone, could be dowloaded by a victim, trying to hide infotmation from an abuser, but could also be installed by an abuser to hide a surveillance app downloaded on a victim's phone. If the clustering were to be succesfull, and we could clearly identify abusers' and victims' devices from clear categories, we might be able to analyze the more ambiguous

categories to try to better understand the landscape.

0.4 Results

Our original results either found inconsistencies in our understanding of the dataset or provided insights into what possible use cases the applications can have. Current findings show how hard it is to automotize this problem, while also highlighting the limitations of previous research efforts of the IPV team.

0.4.1 Establishing Ground Truth

Through qualitative analysis, 19 apps were determined to be malicious enough to be added to the seed set for the algorithm. Moreover, from the 51 apps identified, we saw that 15 apps were not in the scanner that the IPV team developed earlier. Of those 15, 7 apps were identified as malicious enough to be ground truth apps. Therefore, 7 of the 19 apps were not in the scanner built by the IPV team. Once this was discovered, they were added to the list of apps for the scanner.

0.4.2 Codebook

The final codebook consisted of 10 categories (e.g., surveillance, harassment, spoof), with the surveillance category being the most common (372 apps), followed by spoofing (115 apps). This is somewhat expected, since the seed set

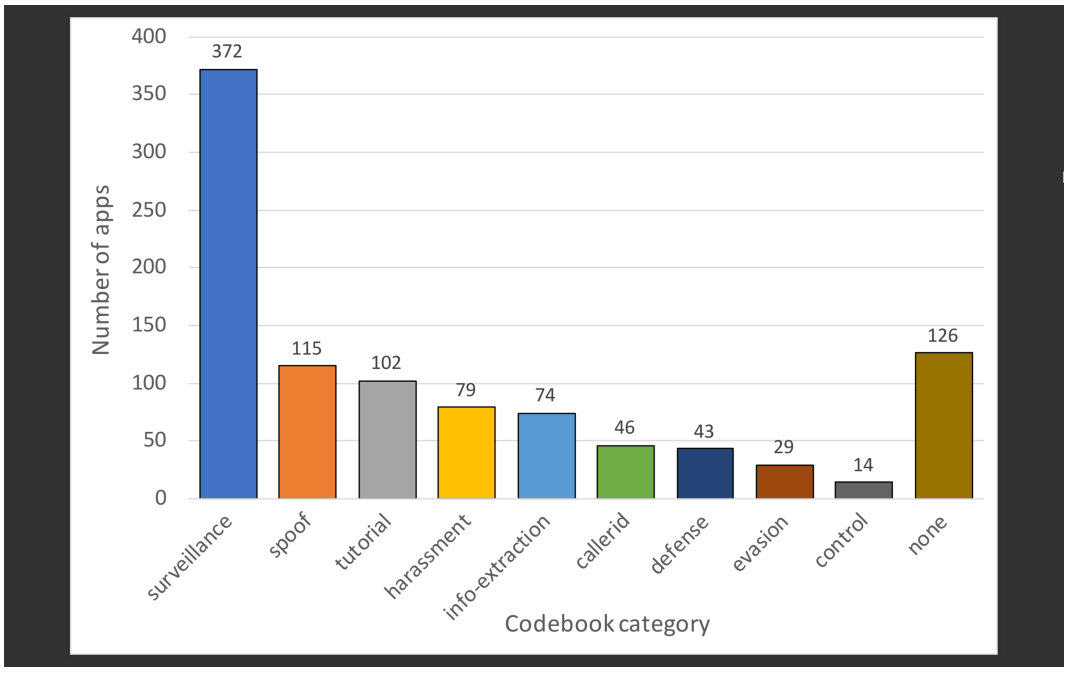

Figure 2: The number of apps in each high level category of the taxonomy. Note that these categories were divided into subcategories to capture more specific IPV purposes.

for Norton's algorithm was primarily composed of surveillence apps found by the IPV team at Cornell Tech. Within the 10 high-level categories, we created a total of 56 sub-category codes (e.g., surveillance:location, harassment:social-media, and spoof:SMS), with the surveillance subcategories being in the top 3 most common subcategories: surveillance:social-media (105 apps), surveillance:location (90 apps), and surveillance:thorough (90 apps). Some interesting findings were common categories like 'tutorial' which were mostly hacking tutorials, helping users to hack someone else's phone or computer; defense apps, mostly used by victims to prevent surveillance or spoofing; or spoof apps, used to identify numbers calling anonymously or easily generate new numbers.

0.4.3 Analyzing Timestamps

Our goal was to see whether there was some patterns behind the installations of malicious apps. To do this, several visualizations were made. Since we were interested in how the installation times compared to each other, we first built a histogram of time deltas for malicious apps, the differences between timestamps of consecutive malicious apps on a single device.

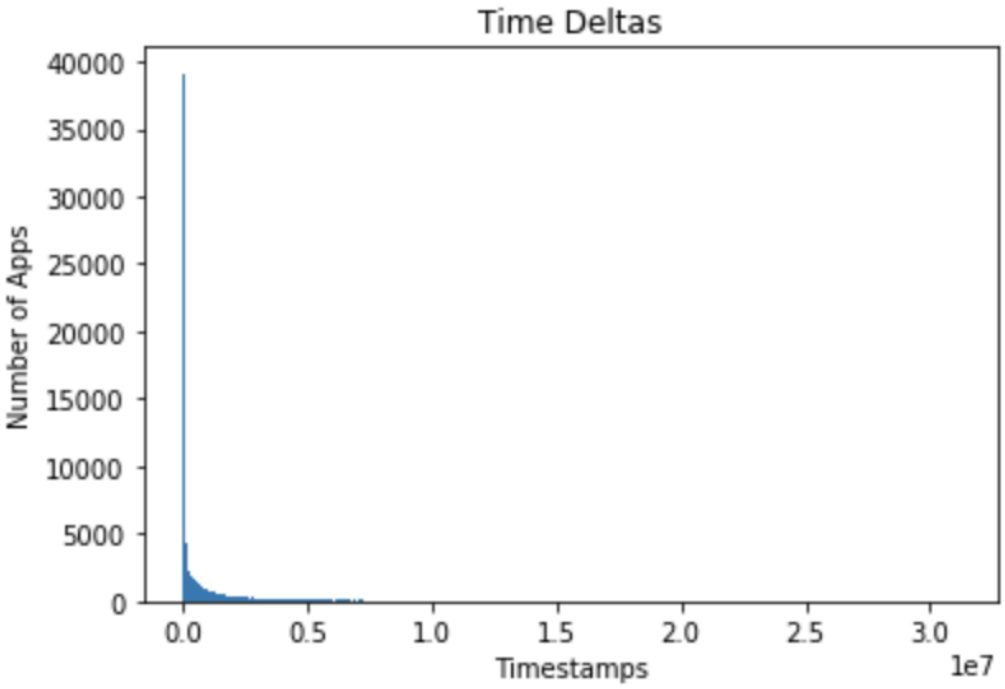

Figure 3: A histogram of the time deltas of consecutive malicious apps on devices. The timestamps are binned by day. Upon inspection of the number of app pairs that have deltas as 0, we conclude that a majority of app pairs have time deltas as 0.

Notice here that most of the time deltas were 0, something that seemed very odd. This indicates two possibilities. Either a lot of timestamps are 0 and so we have a lot of differences between 0s or there are a lot of timestamps that are equal to each other and thus cancelling out when we create our deltas. We also

wondered whether this phenomenon could be found between all apps, not just malicious ones.

We pivoted away from making histograms and made time series charts to analyze the data. Randomly choosing 30 devices, we made time series charts of both time deltas and time stamps for both all apps and malicious apps.

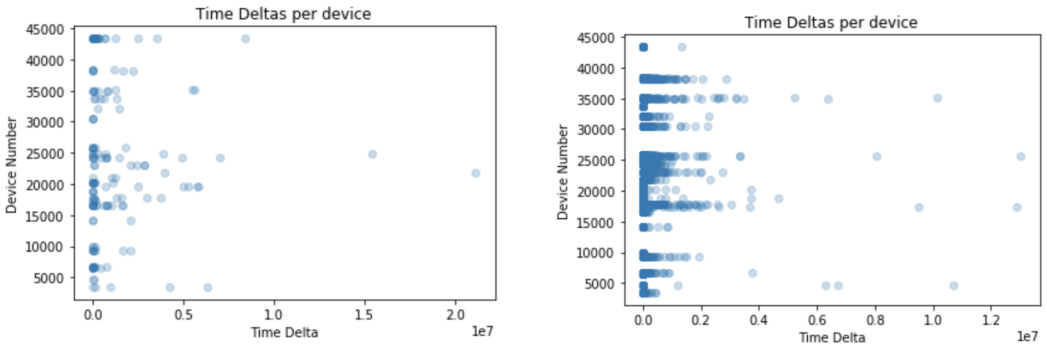

Figure 4: **Left:** A time series chart of time deltas for malicious apps. **Right:** A time series chart of the time deltas for all apps. Each row of dots represents a device. The darker a point is, the more app pairs have that time delta.

Notice that the time deltas for both images cater towards low values or 0. Thus, we are able to conclude that whether we are exploring all applications or malicious applications, these time deltas are clustering towards low numbers. Therefore, we are left with the question of whether these time deltas are generated by 0s in the dataset or equal values. We explore that with the next couple of images.

We see here that the time series of the timestamps confirms that many apps had equal timestamps, which is why we had so many 0s for deltas. This supports the idea of batched scans of the security app provided by Norton. Thus, we concluded that using the timestamps as a proxy for installation was a poor choice.

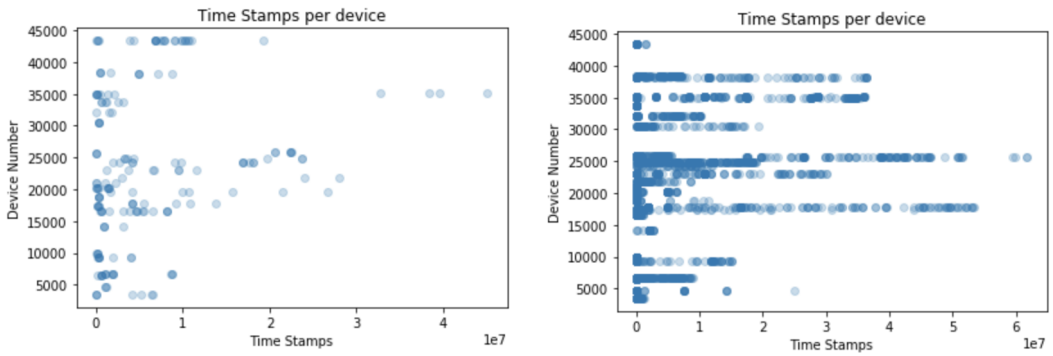

Figure 5: **Left:** A time series chart of timestamps for malicious apps. **Right:** A time series chart of the timestamps for all apps. Each row of dots represents a device. The darker a point is, the more app pairs have that time delta.

0.4.4 Utilizing Search Engine Optimization (SEO)

In order to prove the need of a thorough keyword research and the purchase of the necessary tools to create it, a small study was done to show the potential that was being missed in the previous methods used by Chatterjee et. al.[1]. A single app from the top 1000 apps was used, GuestSpy, to grab the list of its competitors and analyze them. A list of competitors in the SEO setting simply means websites (or apps) that rank high for the same search terms as the website being analyzed. From SemRush we got the list of competitors for GuestSpy and analyzed the first 60. Out of those, 20 were relevant IPV apps, the rest were showing informational sites, blogs, or affiliate websites. Out of these 20, 8 were free and downloadable (and were downloaded and checked manually). Of those 8, 7 were not found in the previous list created by the IPV team at Cornell Tech, and none of them were found in the top 1000 apps list provided by the vendor. This proved that indeed there is a considerable part of the keywords that were not being considered using the snowball search method and that a more exhaustive method was needed to provide a full map of keywords that

would better represent the IPV scene.

To date, the IPV team has now purchased a tool to help with Search Engine Optimization tecniques. To our knowledge, the tool is being used to help generate keyword lists for foreign languages.

0.4.5 Co-Occurreneces

We chose two applications that the vendor qualitatively thought were interesting, and found the top 10 apps that had the highest PMI scores shared with the three chosen by the vendor. We describe what we can say and potential questions that can be asked.

Blue Whale

Note in the graph below the sheer diversity of applications.

Although manual checks need to be done to check whether the use cases of such applications are truthfully implied by their titles, the diversity of apps suggests that the Blue Whale app is something that co-occurs with many different types of apps.

More disturbingly however is the app with the highest PMI on the graph: The Blue Whale Game. This suggests that there could be an app with similar functionality or, even worse, an app that is the same as the Blue Whale app, but under a different title and app-id. If this is true, it has serious consequences on how apps must be looked at in the future and over time because this might

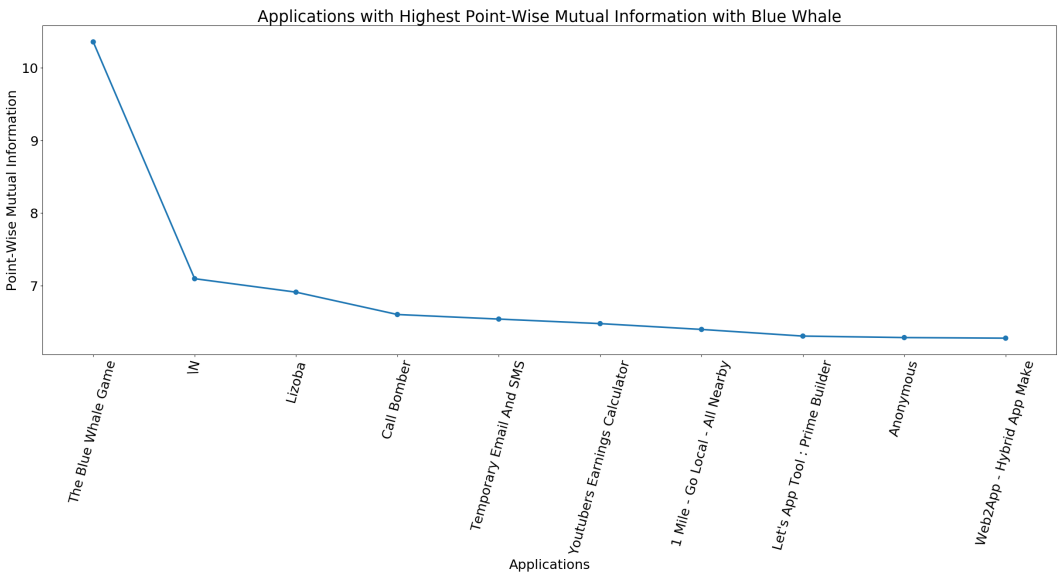

Figure 6: The top 10 applications in respect to the PMI scores against Blue Whale.

mean developers are trying to overcome the shuting down of apps by publishing similar ones under different app-ids.

Edit Website Content Prank

In this graph, we not only observe a strange diversity of apps, but also apps that may or may not be malicious.

Some of the interesting apps that have high PMI scores are potentially IDEs and actual webpage editors. These high PMI scored apps suggest that the Edit Website Content Prank app may not be malicious. These are the types of insights we can glean from PMI scores: if the app is associated with apps that are not malicious, it may not be malicious. However, further research and evidence needs to be collected before we can conclusively say so.

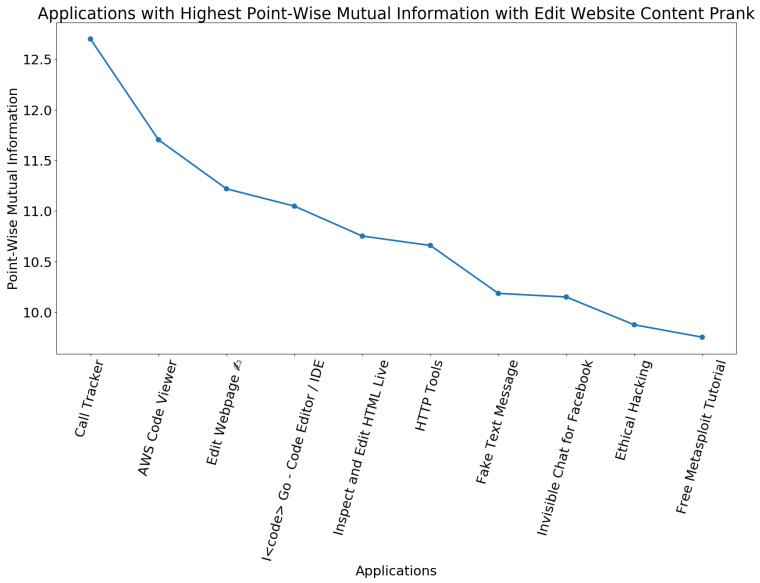

Figure 7: The top 10 applications in respect to the PMI scores against Blue Whale.

0.4.6 Data Collection

The webscrapper effectively searched for apps on the APKPure site. However, the scrapper proved ineffective for searching malicious apps under keywords such as “Track My Girlfriend”. Searching the APKPure site with such keywords only produced benign apps, even though using Google to search for “Track My Girlfriend APKPure” ranked in the first results page the APKPure page for that app. This held true for other app search engines, such as Google Play Store. We conjecture that previous work from the IPV team resulted in app search engines blocking such malicious content.

We tried to get around this in various ways. First we tried querying Google, Bing, and DuckDuckGo. However, due to pricing and the ability of search engines to detect bots, we were not able to use them to collect data. Moreover, professional tools such as Screaming Frog offered ways to grab website infor-

mation, but since we did not have an effective way to do a proof of concept as to the efficacy of Screming Frog for our project, we did not pursue this path. Thus, our biggest problems collecting the data were cost, time, and lack of knowledge of whether professional tools provided the information we needed.

We eventually decided to use a more limited dataset. We scaled down from the over 4,000 apps we had to a smaller set of 1090 apps. These 1090 apps had market place data that was used in Chatterjee et. al. to create a similar feature space as found in that paper. Norton provided the data for the unlabelled apps. Norton has over 3,000,000 unique apps in their database, of which 20% of the data had market place data. We were handed 391,805, which is about 10% of their unique app dataset. However, after filtering out foreign languages using langid and filtering out duplicates in the dataset, we had 80,476 apps available, around 2.5% of the apps available in the initial dataset.

0.4.7 Classification

	Logistic Regression	Random Forest	Naive Bayes	SGD Classifier
Precision	0.68	0.60	0.41	0.68
Recall	0.52	0.38	0.22	0.61
F-1 Score	0.57	0.42	0.24	0.64

Model selection scores for Logistic Regression, Random Forest, Naive Bayes, and SGD Classifier. Each of these can be referenced in the documentation of sci-kit learn.

The most important metric during model selection was recall, as it would make sure that a bad app would not be classified as benign. Thus, we can see that the SGD Classifier from sklearn out-performed logistic regression. After

hyperparameter tuning, the SGD classifier had 0.66 precision, 0.63 recall, and 0.64 F1-Score. Hyperparameter tuning was done with recall as the objective metric, thus these results are inline with what was trying to be achieved.

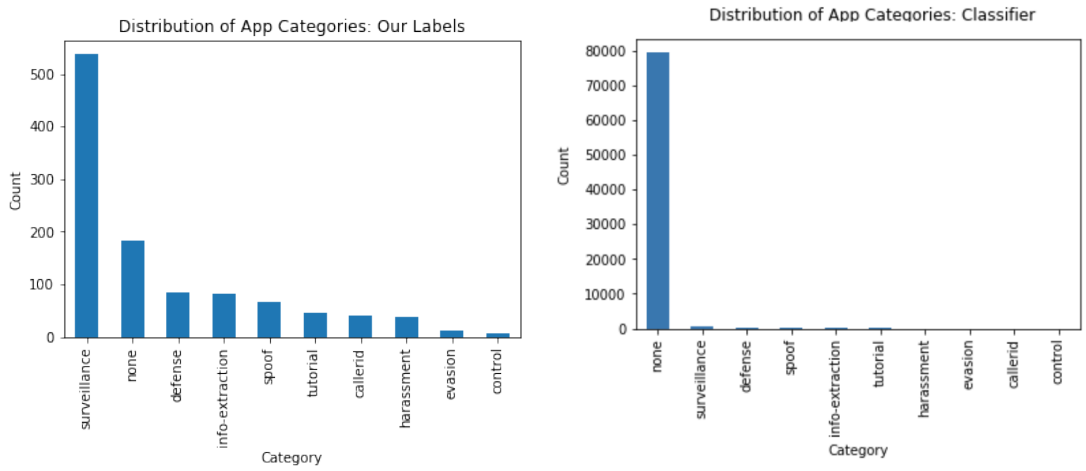

Figure 8: These represent the distribution of classes in our training set and our testing set. **Right:** The distribution of hand labeled apps that had marketplace data. This worked as our training data for our classifier. **Left:** The distribution of the classes that our unlabeled data obtained from our best trained classifier.

However, after labeling the unlabelled apps, we found that it seems the model memorized the training distribution instead of actually learning from the features. There are two possible reasons for this. The class imbalance in the training data set was quite pronounced, with around half the data set containing surveillance apps. Moreover, the distribution of the training data set was known to be different than the real world distribution of apps. As we can see from the classifier's labels, the apps were primarily labeled as none. This makes sense, as most apps are benign. However, the ordering the apps is so similar to the ordering of the training class distribution that we believe there was some overfitting.

We did a balanced sample of 100 data points (10 of each class) from the unla-

beled data after the classifier labeled the data points. We did this because when taking a random sample from the unlabelled data, the chances that the apps are benign is too high. Thus, if we wanted to see how the classifier was doing on all apps in a timely manner, we opted for a balanced random sample instead of a random sample or a stratified sample. Here, we saw some disappointing results.

Overall accuracy of our model, meaning how many labels from the model matched up with manual labels by us researches, was 0.28. We can think of this metric as relating to precision, which is far lower than what we got in cross validation and hyperparameter tuning. Moreover, accuracy against four of the classes (Harassment, Evasion, Control, Caller-Id) were 0. This tells us that the model definitely overfit in the way of memorizing the training distribution, and gives credance to us not having enough training data. The lack of training data is due to the way we had to collect data for feature engineering, as described in the data collection section.

Inspired by Chatterjee et. al. again, we tried to use sklearn's feature selection algorithms along with PCA to see if we could get better results. The results are quite dismal, as reported below.

	Logistic Regression	Random Forest	SGD Classifier
Precision	0.51	0.40	0.46
Recall	0.30	0.31	0.31
F-1 Score	0.33	0.33	0.33

Model selection scores for Logistic Regression, Random Forest, and SGD Classifier, after using PCA and best k features. Each of these can be referenced in the documentation of sci-kit learn. Note that Naive Bayes cannot operate on negative values, and thus was dropped.

This provides even more evidence that there is a lack of data in the training data set. Thus, further feature engineering efforts may not increase performance. However, one other reason that the classifier did poorly was because of the taxonomy from Roundy et. al.[4] Although the taxonomy was developed rigorously, the intention of the labeling could have resulted in granular classes. Thus, we decided to use Latent Dirichlet Allocation [5], a popular topic modeling algorithm, to see if the apps clustered the way the taxonomy assumes.

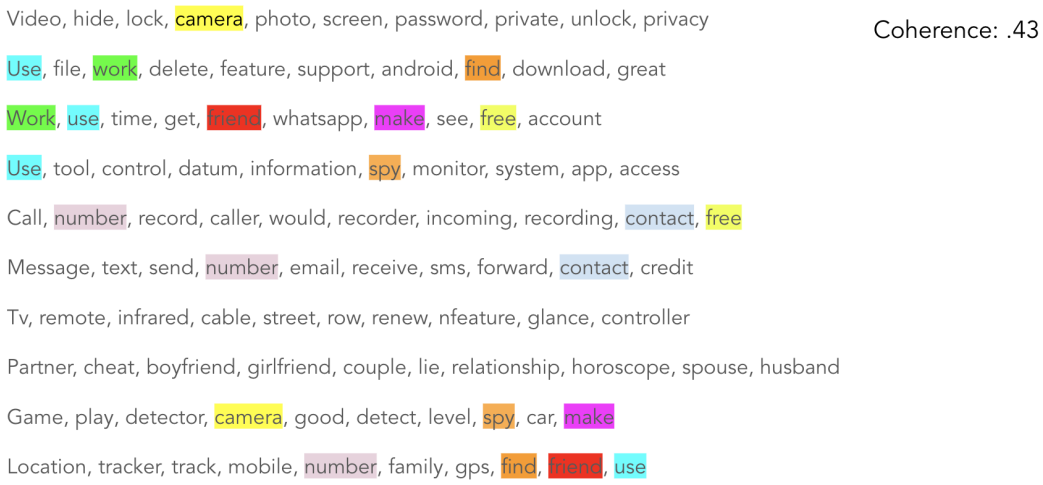

Figure 9: A representation of the topic modeling efforts. Each line of 10 words represents a topic. The highlighted words represent words that appear in mupltiple clusters. The coherence metric, using PMI scores, is also presented.

Several interesting points to note about the clustering via LDA. First, we notice that many words are overlapping. Second, we notice that the coherence score, that is related to the c_v metric in the gensim package, is fairly low (.43 with optimal score at 1). Thus, there are too many clusters.

This does not mean that the taxonomy built made too many clusters at a high level. It could also mean that given the data we have, we cannot accurately classify into the taxonomy. Thus, further data collection efforts should be done.

0.4.8 Clustering

In the first attempt to run k-means on the dataset, with count of labelled apps for each category for each device as the feature vector, and after running the elbow method and discovering that a good k would be between 3 and 4 clusters, the results showed three clusters were all heavily described by the 'none' category. In addition to 'none' being the most important feature for all 3 clusters, the next most important features followed the same distribution of labelled apps we talked about in the Data Collection section: 'none' was followed by 'surveillance' followed by 'info-extraction'. The same result showed for 4 clusters. In order to improve these results, we tried some alternatives.

Initially we tried to normalize the counts by using percentage of apps instead of counts, so each device, under each categroy, would have

$$\%count = \frac{count\ of\ labelled\ apps\ for\ category}{total\ installed\ apps\ in\ device}$$

This approach yielded a similar result to the first run.

On a second attempt, the category 'none' was removed. In this case the next most predominant categories showed up as the most important features for the clusters. So cluster 1's most predominant feature was 'surveillance', cluster 2's was 'defense' and cluster 3's was 'spoof', ('defense' and 'spoof' categories have a similar amount of labelled apps to 'info-extraction'). The 'none' is a category that should be removed regardless. Firstly, a device that has the same number of surveillance apps as a second device, but has double the amount of none apps, should not be clustered in a different cluster. The 'none' apps doesn't say anything about the profile of the device. In addition, a device that is truly not

an IPV device (neither victimm nor abuser) would simply have 0s as counts for all categories, and would therefore be classified in its own 'none' cluster.

'callerid'	'control'	'defense'	'evasion'	'harassment'	'info-extraction'	'spoof'	'surveillance'	'tutorial'
0.0105	0.0026	0.0000	0.0103	0.0029	0.0376	0.0426	0.0000	0.0043
0.0882	0.0140	0.1910	0.0485	0.0258	0.1708	0.1953	1.2821	0.0367
0.0363	0.0148	1.2164	0.0404	0.0170	0.1425	0.1567	0.0482	0.0321

Figure 10: The three clusters formed removing the 'none' category

In a fourth attempt, we tried to balance the dataset of labelled apps. We took 77 apps for each category (except 'control' since it only had 18) and tried clustering with count of apps for each category. In this case, and since the labelled dataset was even smaller, all devices had either none, 1, or 2 labelled apps, which made the model cluster the device as the category of that 1 app in most cases. It is important to note though, that although this says nothing about the clusters that resulted, it was the first time that the clusters did not follow the labelled dataset distibution, but rather 'caller-id' and 'defense' were the most important features for the firt two cluster (the third was a 'none' cluster).

'callerid'	'defense'	'evasion'	'harassment'	'info-extraction'	'none'	'spoof'	'surveillance'	'tutorial'
0.0000	0.0000	0.0113	0.0022	0.0033	0.0060	0.0046	0.0045	0.0049
1.2636	0.0091	0.0575	0.0164	0.0442	0.0310	0.0447	0.0679	0.0264
0.0467	1.0625	0.0667	0.0215	0.0488	0.0662	0.0504	0.0672	0.0767

Figure 11: The three clusters formed using a balanced laballed dataset

This shows, as expected, that the labelled dataset plays a big role, and it reinforces our belief that an app classifier is necessary for any kind of device profiling.

In a last attempt, instead of counts or percentages, the sum of hygiene scores

given by the algorithm provided by Norton were used. Higher scores denote apps appearing in more devices and appearing in devices with many high scored apps. In this way, abuser apps would probably have a higher score than a dual use app, or a benign app, assuming the efficacy of the algorithm is high. Since the seedset used initially to run this algorithm was heavily skewed towards surveillance apps, two of the three clusters showed surveillance as their most important feature and 'none' for the third cluster. This again does not surprise us, since 83 of the top 100 scored apps were surveillance apps.

As expected, no real results could be concluded from this modelling, except perhaps the reasurance that we need more labelled apps (if not all) to have a rich enough dataset to attempt clustering devices. A well perfoming app classifier is therefore necessary in order to say anything about whether device profiling is at all possible.

0.5 Discussion & Future Work

The work presented here is beyond conclusive. The results reveal more questions that need to be answered rather than giving answers to the research questions originally asked. Although we were not able to ascertain any patterns of malicious installations, we were able to better understand the dataset given by the vendor. Moreover, the data given allowed us to construct a taxonomy for the ecosystem of applications that could potentially exist in the IPV space. Some findings show that the taxonomy is a bit too granular, but there are many limitations to this finding, including data collection practices. Moreover, our findings suggest that making conclusions using incomplete data is insufficient

and cannot accurately describe the space.

The biggest takeaway that will be discussed is data governance, as it was a big contributor to our findings and our work. Teams looking to build on this work should be aware of the nuanced ways data governance can affect, at scale, the necessary workflows.

0.5.1 Classification to Clustering

The original theorized pipeline would have labelled all the applications existing in the Norton database, then using those labels, along with other information such as app text data, PMI scores, etc., as a feature space to cluster devices. This clustering of devices hopefully would have revealed patterns in use, and thus would have brought us one step closer in identifying victims and abusers.

However, due to failures in data collection and limited data resources, we were not able to fulfill the original criteria, such as each device having all labeled apps. Thus, many of our findings could never be generalized, as it would not characterize a device in full. It would be irresponsible to define a device as an abuser's without knowing the full range of apps on the device. But thanks to our work, we know this definitively. Those looking to build on our work should take care to make sure the dataset is exhaustive and comprehensive. Without those measures taken, much of our weaknesses, such as distribution memorization of the training data, could taint results.

0.5.2 Data Governance

We have over 5+ years working in technology companies between us. When working on data science projects, clarity in communication from those querying data is essential to successfully execute on products in a timely fashion. Accurate explanations of data sources, consistent file formats, and basic statistical information all help in the data transfer process.

The lack of data governance when working with Norton's representative was unfortunately detrimental to a lot of the work planned for the past year. In hindsight, establishing clear data governance principles around file types, data documentation, consistent schema development, timelines on data processes, and coherent communication would have resulted in a smoother collaboration. Many times misleading information about the data (i.e. the data contained unique apps, but the data had many duplicates), vague or incorrect descriptions of datasets (lack of schema or feature description), lack of transparency about how data was collected (incorrect description of how timestamps were sourced, until analyses were done), and lack of transparency of available data. Along with this list, the Norton representative made various insights/hypotheses about the data throughout the project. Appropriately, we questioned these insights, so we could understand how the data was feeding into them. At times it would take months to get an answer, and some insights were never given rigorous justification by Norton's representative. This resulted in elongated project timelines, and unfortunately most of our efforts were put into fixing errors rather than making progress.

The work of a data scientist includes the entire data pipeline: collecting data, exploring datasets, validating assumptions, establishing proper datapipelines,

creating documentation, building models when appropriate, and more. However, when miscommunication occurs in this process, it disables the data scientist's work, as they are operating on a different set of assumptions than what is true. In addition, different data scientists receiving this misrepresented data may even be working on different assumptions altogether, resulting in inconsistent results from members of the same team. Thus, establishing improved communication and proper data governance would have alleviated many of our issues with Norton's representative.

0.6 Conclusion

Our work shows that using the learnings from an app classifier to cluster devices, requires that each algorithm does not suffer from data compromising issues. If this work were to move forward, we strongly recommend starting at the data collection phase and getting good data to build a classifier for apps based on the IPV team's taxonomy; discerning whether this taxonomy is too exhaustive or comprehensive, and thus trying to cluster devices for patterns.

0.7 Bibliography

1) Rahul Chatterjee, Periwinkle Doerfler, Hadas Orgad, Sam Havron, Jackeline Palmer, Diana Freed, Karen Levy, Nicola Dell, Damon McCoy, and Thomas Ristenpart. "The Spyware Used in Intimate Partner Violence.; IEEE Symposium on Security and Privacy (Oakland 2018).

2) Diana Freed, Jackeline Palmer, Diana Minchala, Karen Levy, Thomas Ristenpart, and Nicola Dell. "'A Stalker's Paradise': How Intimate Partner Abusers Exploit Technology." ACM Conference on Human Factors in Computing Systems (CHI 2018).

3) Diana Freed, Jackeline Palmer, Diana Minchala, Karen Levy, Thomas Ristenpart, and Nicola Dell. Digital Technologies and Intimate Partner Violence: A Qualitative Analysis with Multiple Stakeholders. Proceedings of the ACM on HumanComputer Interaction: Volume 1 Issue CSCW, November 2017. Article No. 46.

4) Kevin Roundy, Paula Mendelberg, Nicola Dell, Damon McCoy, Daniel Nissani, Thomas Ristenpart, and Acar Tamersoy. The Many Kinds of Creepware Used for Interpersonal Attacks. IEEE Symposium on Security and Privacy (Oakland 2020).

5) David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3 (March 2003), 993-1022.