Device Graphing by Example

Keith Funkhouser
comScore
kfunkhouser@comscore.com

Matthew Malloy
comScore
mmalloy@comscore.com

Enis Ceyhun Alp
comScore
ecalp@comscore.com

Phillip Poon
comScore
ppoon@comscore.com

Paul Barford
comScore, University of Wisconsin
pbarford@comscore.com

ABSTRACT

Datasets that organize and associate the many identifiers produced by PCs, smartphones, and tablets accessing the internet are referred to as *internet device graphs.* In this paper, we demonstrate how measurement, tracking, and other internet entities can associate multiple identifiers with a single device or user after coarse associations, *e.g.,* based on *IP-colocation,* are made. We employ a Bayesian similarity algorithm that relies on examples of pairs of identifiers and their associated telemetry, including user agent, screen size, and domains visited, to establish pair-wise scores. Community detection algorithms are applied to group identifiers that belong to the same device or user. We train and validate our methodology using a unique dataset collected from a client panel with full visibility, apply it to a dataset of 700 million device identifiers collected over the course of six weeks in the United States, and show that it outperforms several unsupervised learning approaches. Results show mean precision and recall exceeding 90% for association of identifiers at both the device and user levels.

CCS CONCEPTS

• **General and reference** → *Measurement*; • **Mathematics of computing** → *Bayesian computation*; • **Information systems** → *Online advertising*; • **Computing methodologies** → *Supervised learning by classification*;

KEYWORDS

Internet measurement, device graph, naïve Bayes

ACM Reference Format:
Keith Funkhouser, Matthew Malloy, Enis Ceyhun Alp, Phillip Poon, and Paul Barford. 2018. Device Graphing by Example. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219852

1 INTRODUCTION

Web publishers, advertising networks, and measurement and analytics companies collect diverse information from devices as they access internet services. The information they collect is used in a variety of ways including targeted advertising, reporting, auditing, and fraud detection. There are two primary types of identifiers used by third-parties to track user activity *across* the internet: OS-level advertising identifiers (*e.g.,* Apple's Identifier For Advertisers (IDFA) and Android's Advertising ID) and third-party web cookies. We refer to both types of identifiers generically as *IDs*.

From the perspective of a third-party measurement, advertising, or tracking entity, the online activity of a single user is rarely captured by a single ID, primarily for two reasons. First, users increasingly access the internet from multiple devices (*e.g.,* a smart phone and a laptop), each of which presents a unique ID with no immediate indicators that the IDs are related. Learning the relationships between IDs across devices shared by the same user is termed *cross-device identification.* Second, a single device presents multiple IDs depending on how and when the internet is accessed. A smartphone accessing resources on the internet may present one cookie when using the default web browser, a different cookie for each web browser embedded in an app (*e.g.,* Facebook's embedded browser), and an advertising ID when using other applications. Often this *intra-device ID multiplicity* is by design, with the aim of limiting and preventing tracking. Apple's latest release of iOS 11 (September 2017) includes "Intelligent Tracking Prevention" [4], which both actively deletes third-party cookies and creates *first-party-specific third-party cookies.* The result is an increasing number of increasingly ephemeral IDs from a single device. We term the problem of associating multiple IDs with a single device *intra-device identification.*

Datasets that capture relationships between IDs at scale are called *internet device graphs.* Over the past several years, device graphs have become ubiquitous in the digital advertising ecosystem; facilitating targeted advertising, content customization, and tracking of media consumption. Approaches for generating internet device graphs are typically classified as *deterministic* or *probabilistic.* Deterministic graphs refer to approaches that directly relate login, email, or other personally identifiable information to IDs. Probabilistic graphs are derived using algorithms that ingest data such as IP-address information, geo-location, or other information to *infer* an underlying relationship between IDs.

In this paper, we address the problem of relating multiple IDs to the same device or user. Our work complements and extends techniques for probabilistic internet device graphing such as [24].

We describe a learning algorithm for grouping related IDs after coarse groupings, such as household-level groupings (*i.e.,* IDs that belong to users of the same residential household), are made. The approach is generalizable in that it can be used to address both cross-device and intra-device identification (*i.e.,* create device- and user-level groupings) depending on the features employed and the training data available. The learning algorithm proceeds in two phases. First, a naïve Bayesian similarity scoring algorithm relies on labeled example pairs of IDs and their associated telemetry. The pairs of IDs are *labeled* in that a ground truth indicator identifies the IDs as belonging to the same group (*i.e.,* device or user) or different groups. The features employed are chosen to discriminate between different groupings. In the case of device-level groupings, features containing information specific to the device, such as user agents (UA) and screen resolution, are utilized by the algorithm. For user-level groupings, behavior-based features specific to a user, such as a list of recent domains and mobile applications, are used. The algorithm scores the likelihood that an unlabeled pair of IDs belong to the same group by comparing features to the labeled examples. A high scoring pair of IDs corresponds to two IDs from the same group. After pair-wise scores between IDs are established, the IDs can be clustered into small groups that correspond to individual users and devices using community detection algorithms. We apply a modified *Louvain Modularity* community detection algorithm [8] to the pair-wise scores, producing small groups of IDs representing devices or users.

We demonstrate our methodology at internet scale with an immense dataset[1] from comScore's *digital census network*. comScore's census network consists of web page, advertisement, and application tags deployed by publishers across the internet. The tags generate server logs of client requests to publisher web pages, requests for advertisements, and applications that request resources on the internet. This dataset, collected over a period of 6 weeks in the United States, includes 700 million IDs, hashed IP addresses, and supplemental signals including the UA and screen resolution. Implementations of the algorithms in a Hadoop environment process the entire dataset in hours. We report the results of the application of our method to the digital census network data in Section 3.2.

We train and validate our methodology using data from *user panels* that are operated by comScore. These panels are realized through specialized monitoring software and customized in-home hardware, and are opt-in (*i.e.,* users provide explicit permission prior to installation). The user panels provide training data that includes a unique device hardware identifier (*e.g.,* its Media Access Control (MAC) address) and the IDs associated with that device. Using leave-one-out cross-validation, we generate receiver operating characteristic (ROC) curves that capture the error rates of the approach. The results are compared to unsupervised classification approaches and greatly outperform them, highlighting the ability of our methodology to accurately associate pairs of IDs that belong to the same device or user. After validating the pairs of IDs, we validate the clustering of IDs. We generate precision/recall curves for ID clustering based on a modified Louvain Modularity algorithm. The results show mean precision and recall rates above 91% are possible at the device level. At the user level, precision and recall rates exceed 94%. However, this analysis belies the difficulty of user-level grouping, illustrated by the comparatively lower true positive and true negative rates observed in leave-one-out cross-validation testing (Figs. 3 and 4). Section 3.3 details the validation of our approach.

Lastly, in light of a rapidly changing digital advertising, tracking, validation and measurement ecosystem, we discuss privacy and ethics in learning device- and user-level relationships between identifiers at scale. We note that our consideration of this problem respects the privacy settings of a device, including exclusion of IDs associated with incognito browsing and rejected third-party cookies. It is not our aim to associate IDs that are actively reset by a user. Instead, we aim to group IDs that are generated by the multiple ways in which a single device or user accesses the internet. We advocate these properties as key characteristics of a privacy-respectful device graph dataset[2].

In summary, this paper makes the following contributions. First, to the best of our knowledge, this is the first study of the problem of grouping multiple IDs *to an individual device or user* at scale. Second, we describe a general ID–to–device and ID–to–user matching method based on Bayesian learning and Louvain Modularity. We demonstrate the efficacy of our approach on unique datasets of immense scale and report on basic characteristics of device- and user-level groupings.

2 METHODOLOGY

2.1 Preliminaries

While our methodology applies to internet data at scale, our starting point is coarse groupings of related IDs, such as the *household-level* groupings described in [24]. Coarse groupings, which we refer to generically as *cohorts*, can be created from internet measurements in a variety of ways. For example, cohorts can be comprised of all IDs observed on the same residential IP address over some period of time, or all IDs that access a particular web resource (*e.g.,* a web site) over some period of time. Ultimately, cohorts consist of IDs that are predisposed to belong to a fine-scale grouping of interest. In this context, cohorts are coarse groupings, comprised of a number of smaller fine-scale *device*- and *user*-level groupings. Our objective is to divide a cohort into these fine-scale groups.

While the methodology in this paper can be used with a wide range of cohorts as input, the evaluation and implementation builds upon [24], which creates cohorts that approximate residential households. More specifically, [24] begins by building a large graph data structure by creating edges between IDs observed on the same IP address at similar times, a technique termed IP-colocation. IP-colocation occurs when devices share an IP address, commonplace when connecting to the internet via the same WiFi router. After creation of the graph data structure, community detection algorithms are applied to group the IDs into cohorts of different sizes, and the cohorts corresponding to residential households are validated. Both steps are highlighted in the top half of Fig. 1 (outside the dotted box). The graph and the groupings reported in [24] consist of more than 700 million IDs and 100 million cohorts.

[1]We plan to make an anonymized subset of this data available to the research community on publication.

[2]Users can reset or block IDs by adjusting OS and browser privacy settings. Privacy-related concerns of user tracking are treated in Section 5

Before describing our methodology, we formalize notation. A weighted graph $G = (V, E)$ is a set of nodes V and a set of weighted edges E. A weighted edge $e \in E$ is a two element subset of V and a weight: $e = (i, j, w) \in V \times V \times \mathbb{R}$. In a *device* graph, a node $i \in V$ is an ID (*e.g.,* a web cookie or advertising ID). Cohorts are a set of one or more nodes, $C = \{i, \dots\} \subset V$. The set of cohorts is denoted by $\mathscr{C} = \{C_1, C_2 \dots\}$ and $\mathscr{C}$ is a partitioning of V, *i.e.,* $C_m \cap C_n = \{\}$ for any $m \neq n$, and $\bigcup C_m = V$.

Our objective is to subdivide cohorts into smaller groups of IDs corresponding to devices and users based on information, or features, pertaining to each ID. Let X_i denote the features associated with node i and let U be a set of one or more IDs that represent a device or a user. We assume U is a non-overlapping subset of one cohort: $U = \{i, \dots\} \subset C \subset V$, and $U_m \cap U_n = \{\}$ for all $m \neq n$.

The goal of our work is to answer the following questions: Given as input *(i)* a set of cohorts and *(ii)* feature sets X_i associated with each ID i, how can one subdivide the cohorts into groups of IDs corresponding to devices and users? Given a cohort $C = \{i_1, i_2, \dots\}$ and features $X_{i_1}, X_{i_2}, \dots$, how can one learn $U_1, U_2 \dots$?

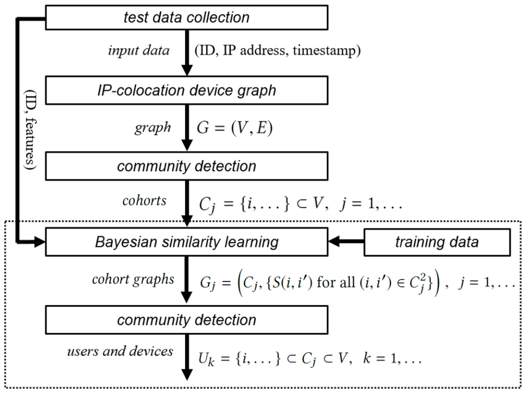
test data collection
input data
(ID, IP address, timestamp)
IP-colocation device graph
graph
$G = (V, E)$
community detection
cohorts
$C_j = \{i, \dots\} \subset V, \ j = 1, \dots$
(ID, features)
Bayesian similarity learning
training data
cohort graphs
$G_j = \left(C_j, \{S(i, i') \text{ for all } (i, i') \in C_j^2\}\right), \ j = 1, \dots$
community detection
users and devices
$U_k = \{i, \dots\} \subset C_j \subset V, \ k = 1, \dots$

Figure 1: Block diagram representing a combination of IP-colocation-based device graphing described in in [24] and the user/device identification method proposed in this paper (highlighted by dotted box).

We address these questions in two steps in the following sections. First, a similarity score for each pair of IDs that share a cohort is calculated. The similarity score between two IDs that share a cohort is denoted by $S(i, i')$, and for each cohort, these scores comprise a weighted graph (denoted by G_j in Fig. 1). The similarity score reflects the likelihood that the IDs share a common grouping, *i.e.,* belong to the same device or user. The second phase of our approach relies on a variant of Louvain Modularity community detection that accounts for negatively weighted edges to group the IDs. The dotted frame of Fig. 1 captures these two steps.

2.2 Naïve Bayes Similarity Scoring

For each *pair* of IDs that share a cohort, a *naïve Bayes similarity score* (NBSS) is derived. We selected *naïve* Bayes as our classifier for similarity scoring due to the sparsity of the feature space, limited training data, and the need for a highly scalable implementation since our application is for very large datasets. The similarity score measures the likelihood that two IDs belong to the same fine-scale group (*e.g.,* the same device or user). The score can be used to determine a binary hypothesis: a large positive score between two IDs is indicative of the same device or user; conversely, a large negative score implies evidence against the IDs belonging to the same device or user. Scoring is based on the features of the pairs of IDs, and whether examples from training data suggest these features are indicative of IDs belonging to the same grouping.

More specifically, consider two IDs i and i', $i \neq i'$. Let $X = \{x_1, x_2, \dots, x_n\} \subset \mathscr{X}$ and $X' = \{x'_1, x'_2, \dots, x'_{n'}\} \subset \mathscr{X}$ be feature sets (or feature bags) associated with devices i and i' and let $\mathscr{X}$ be a countable set. In the case of device-level groupings, x_1 could be a particular UA string, or screen resolution, *e.g.* 1920×1080, observed with ID i. In the case of user-level groupings, x_1 could represent a domain or a URL, for example *xyz.com*, visited by ID i. Let Y be an indicator of the true class: if IDs i and i' belong to the same grouping, $Y = 1$, otherwise $Y = 0$.

The NBSS is based on a proxy for the posterior probability ratio of the two classes. A scaled sum of the log posterior probability ratios from *all pairs* of features x_ℓ, x'_m is calculated. The premise for this expansion is that information pertinent to classification is well captured by pairs of features, one from each device. NBSS is computed as follows:

$$S(i, i') = \frac{1}{\sqrt{k}} \sum_{\ell=1}^{n} \sum_{m=1}^{n'} \log \frac{P\left(Y = 1 | x_\ell, x'_m\right)}{P\left(Y = 0 | x_\ell, x'_m\right)} \tag{1}$$

where k is the number of non-zero terms in the sum, $P\left(Y | x_\ell, x'_m\right)$ is the posterior probability of the class given observation of ID i with feature x_ℓ and i' with feature x'_m. As such, the similarity score involves the posterior probability given *each pair* of features, one from each ID, derived from counts of examples in training data. The leading constant of $1/\sqrt{k}$ accounts for the power law increase in the number of terms as the number of informative feature pairs increase.

The NBSS approach does not assume the prior probability of the classes is available, as the posterior ratio can be replaced by the likelihood function, and a non-uniform prior can be incorporated into the score by adding a universal scaling constant, which we explore in Section 3.3.

The posterior probability ratio of the classes is estimated as the number of *same group* ID pairs with features x_ℓ, x'_m divided by the number of *different group* ID pairs with features x_ℓ, x'_m in the training set. More specifically,

$$\frac{P\left(Y = 1 | x_\ell, x'_m\right)}{P\left(Y = 0 | x_\ell, x'_m\right)} := \frac{N_1(x_\ell, x'_m) + \alpha}{N_0(x_\ell, x'_m) + \alpha} \tag{2}$$

where $N_1(x_\ell, x'_m)$ is the number of *same group* ID pairs with features x_ℓ, x'_m, $N_0(x_\ell, x'_m)$ is the number of *different group* ID pairs with features x_ℓ, x'_m, and α is an additive smoothing constant. The constant is included to smooth the ratio when $N_1(x_\ell, x'_m)$ or $N_0(x_\ell, x'_m)$ is small.

Once the quantity $S(i, i')$ is determined, a threshold can be implemented to classify the pair of IDs, i and i', as belonging to the same group or different groups.

2.3 Unsupervised Approaches

In addition to NBSS, three unsupervised learning approaches were implemented for comparison. In particular, we explored *(i)* an approach based on the size of the intersection of the two feature sets, *(ii)* an approach based on *n-grams*, and *(iii)* an approach that relies on the *IP-colocation* score. All approaches are compared to random scoring, where each pair of IDs is assigned a score $S(i, i') \in [-1, 1]$ uniformly at random.

The approach based on the size of the intersection can be considered a baseline approach for matching feature sets. In this case, the similarity between the pairs of feature sets from the two IDs is defined as the size of their intersection: $S(i, i') = |X \cap X'|$. The second unsupervised approach is based on extracting trigrams from each bag of words and counting the intersection. More specifically, the trigram of a string is the set of all three character sequences occurring in that string. Let T and T' be the trigrams of the feature sets associated with device i and i' (the feature set is converted to a string by ordering alphabetically and concatenating). The trigram score is computed as $S(i, i') = |T \cap T'|/\min(|T|, |T'|)$.

Third, we explore the use of the IP-colocation score defined as $S(i, i') = w_{i,i'}$ from [24] . The IP-colocation score is computed as follows: for each pair of devices i, i' observed on IP k, on day t, a score $w_{i,i'}(t, k) = 1/N_{t,k}$ is assigned, where $N_{t,k}$ is the number of IDs observed on IP k on day t. Scores are summed across IP addresses and days: $w_{i,i'} = \sum_{k,t} w_{i,i'}(t, k)$. The IP-colocation score does not rely on the features required by the other approaches.

2.4 Grouping Based on Pair-wise Scores

The scored pairs of IDs constitute a *weighted graph*, with nodes defined by IDs and weighted edges defined by the score $S(i, i')$. Grouping nodes of a graph, a problem termed *community detection*, is well studied and we appealed to the vast body of work on this problem. We employed a variant of the Louvain Modularity community detection algorithm [31], which groups nodes by optimizing graph *modularity*, a measure of the strength of the community structure.

Since the NBSS can be negative, we use a modified Louvain community detection algorithm that can accommodate both positively and negatively weighted edges [18]. The approach treats the negative and positive edges as separate graphs when computing modularity. The negative graph repels nodes from sharing a community, while the positive graph attracts. In general, Louvain Modularity is an iterative process in which the communities generated from the first step are treated as nodes on the second step creating a hierarchy of communities as output. Our approach terminates after the first step, outputting the finest scale of communities, but otherwise follows exactly the procedure in [18].

Adding a universal constant to the scores controls the size of the output groupings. In the extreme case, when a large negative value is added, all edges become negative and repulsive, and each ID is grouped only with itself. Adding a large positive constant forces all connected IDs to be grouped together. We note that adding a constant to $S(i, i')$ has the same effect as including a prior probability in the computation of the similarity score.

Figure 2 shows a small example cohort to which the NBSS approach and the modified Louvain Modularity algorithm were applied. The cohort consists of 6 IDs and the technique groups these IDs into three devices.

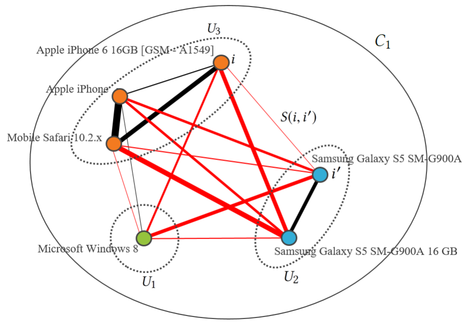
U_3
C_1
Apple iPhone 6 16GB [GSM - A1549]
i
Apple iPhone
$S(i, i')$
Mobile Safari 10.2.x
Samsung Galaxy S5 SM-G900A
i'
Microsoft Windows 8
Samsung Galaxy S5 SM-G900A 16 GB
U_1
U_2

Figure 2: A small example cohort C_1 with 6 IDs. The NBSS learning algorithm is applied to all $\binom{6}{2} = 15$ pairs of IDs. Black edges indicate positive scores, while red edges indicate negative scores. The magnitude of the score is proportional to the width of the edge. After scoring, community detection is applied to group the IDs into U_1, U_2 and U_3.

3 EVALUATION

This section describes the application of our approach detailed in Sec. 2 to an internet-scale dataset. Our goal is to demonstrate the efficacy of the approach, highlight potential shortcomings, and provide a framework for reference.

3.1 Data

comScore’s digital measurement networks are amongst the largest in the world, collecting logs of more than 60 billion internet events per day[3]. comScore collects data through digital tagging of websites, videos, mobile apps, advertisements, web widgets, and distributed content. Tagging refers to placing code alongside web content that directs a device to contact comScore’s web servers. Tagging is implemented through two means: JavaScript/HTML tags and SDK tag implementations. In both implementations, a client device executes the tag code locally and makes a web request directly to a comScore web server. The request is logged and stored in a warehouse in the form of a record.

Training and test data were collected from comScore’s digital census network during a 42 day (6 week) period from August 14, 2017 through September 24, 2017. Data collection was restricted to the US. Over 2 trillion records of internet events in the US were logged over the six week collection period. While the information reported in a record varies depending on the client and the tag implementation, information critical to our study can be divided into two categories: data used to define cohorts, and features utilized by the machine learning approach to further classify IDs.

The first category of data required to define the cohorts used in our study consist of: *(i)* an ID, *(ii)* a hashed IPv4 address, and

[3]In an independent web crawl, 18.19% of the Alexa top 100,000 domains placed third-party cookies of comScore [10], the second largest percent among third-party hosts.

(iii) a timestamp. This telemetry is used to make the coarse associations between IDs using the method described in [24]. Critically, as detailed in Section 2, IP address-based association serves as the starting point for our assessment by reducing the space of possible ID pairs and enabling our method to be applied at scale.

Application of the method in [24] to generate coarse associations resulted in 694 million unique IDs grouped into 102 million cohorts, well aligned to residential households. By considering only pairs of IDs that share a cohort, this resulted in over 3 billion ID pairs to which the Bayesian similarity approach was applied.

3.1.1 Device-Level Features. The features used as input to derive the NBSS were chosen to discriminate between IDs that belong to the same/different groups. In particular, for *device-level groupings*, features describing the hardware, browser, and operating system were chosen. For *user-level groupings*, browsing behavior as indicated by URLs visited and apps used were chosen.

At the device level, we explored inclusion of two types of features: *(i)* the UA string and *(ii)* the screen resolution of the device. A UA string is included in a client HTTP call and in general indicates the type and version of the client web browser and operating system (see [20] for a recent study of UAs). UA strings are available with telemetry associated with web page tags. An example of a typical UA is: *Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36*, which corresponds to Chrome browser running on Windows 10. In the case of an application that sends telemetry via the comScore SDK, the UA is also recorded. While UAs returned by applications (including the comScore SDK) often follow a standard format – for example, an iPhone application – *App/2 iPhone5,2 iOS/11.1 CFNetwork/808.3 Darwin/16.3.0* – the space of UAs returned by applications is vast. We note that our approach requires no classification of the UA into platform and browser information. Instead, training data simply characterizes pairs of UAs as likely or unlikely to belong to the same devices, by learning from example.

The second type of feature employed for the device-level groupings is screen resolution. The screen resolution, reported in vertical pixels by horizontal pixels is collected by both the JavaScript tag and the SDK tag. Screen resolution is standard telemetry included in web requests. We note that mobile devices often return a reduced resolution, hence direct comparison of collected screen sizes may not be suggestive of device-level associations.

The screen size information and the UA for each ID are stored as a set. Values observed over the course of the six weeks are included. A typical feature set contains multiple distinct UAs and/or screen sizes. For example, the feature set associated with an observed ID was: $X = \{$'Mozilla/5.0...', 'App/2 iPhone5,2...',1080x1920, 540x960$\}$. The feature set was limited to the top ten features (by count of days observed) for each ID.

The corpus of test data collected over the six week period consisted of 3,489,655,490 ID pairs corresponding to 740,805,009 distinct IDs and 2,489,359,280 distinct (ID, feature) pairs, implying an average of 3.36 features per ID.

3.1.2 User-Level Features. To learn groupings of IDs that are shared by a single user, the features were chosen to capture aspects of user behavior. For each ID i, the set of top-level domains and applications used was collected. As cookie IDs are primarily associated with web browsing, the feature sets of these IDs were sets of top-level domains on which the ID was observed, for example, $X_i = \{\text{website1.com}, \text{website2.com}, \dots\}$. Advertising IDs, conversely, are available to applications running on a mobile device, including applications which incorporate the comScore SDK. For advertising IDs, a set of applications used by the ID comprised the feature set, for example, $X_{i'} = \{\text{App 1}, \text{App 2}, \dots\}$.

The corpus of test data collected over the six week period consisted of 3,117,687,944 ID pairs corresponding to 694,409,939 distinct IDs and 4,062,302,884 distinct (ID, feature) pairs, implying an average of 5.85 features per ID.

3.1.3 Device-Level Training Data. comScore collects detailed measurements from a client panel (CP) that deploys instrumented routers in each client's home. The routers are gateways to/from the internet and capture browsing statistics for all connected devices in the household. The router also records an obfuscated version of the media access control (MAC) address and the IDs associated with any device that connects to the internet via the router.

The data required to model the posterior class probabilities given pairs of features, $P(Y|x_\ell, x'_m)$, are informed by the CP. More precisely, pairs of IDs that share a universal hardware identifier are known to belong to the same device, hence they can provide example feature pairs belonging to the *same* machine, *i.e.,* $Y = 1$. Likewise, pairs of IDs that *do not* share a universal hardware identifier do not belong to the same device, therefore they present examples of pairs of features belonging to *different* devices, *i.e.,* $Y = 0$. The corpus of training pairs was limited to ID pairs that belong to the same CP household.

After data normalization and cleaning, the training dataset collected from the CP consisted of 88,376 IDs associated with 42,650 MAC addresses, partitioned into 13,068 households. IDs associated with more than one MAC address were excluded from consideration. By considering only the pairs of IDs that belong to the same household, the dataset was used to define 336,087 pairs of IDs. Of the 336,087 pairs, 76,125 corresponded to ID pairs with a shared MAC address, and 259,962 corresponded to ID pairs with a different MAC address.

Combining the 336,087 labeled pairs with their associated features resulted in 1,813,278 distinct feature pairs. If $N_1(x_\ell, x'_m) + N_0(x_\ell, x'_m) \leq 2$, then the feature pair was not included as a valid training example. This reduced the total number of feature pairs in the resulting training set to 354,685. We refer to these pairs as the set of *training pairs.*

3.1.4 User-Level Training Data. A third-party dataset was used as training data for user-level groupings. This dataset consisted of pairs of IDs that are associated with the same user and pairs of IDs associated with different users[4]. The pairs of IDs were limited to those that shared a cohort (a residential household) produced by the methodolgy in [24], resulting in a dataset consisting of 3 million pairs of IDs that belonged to the same users, and 20 million pairs that belonged to different users.

[4]Our terms of use with the third-party prevents us from sharing their name. The third-party uses a variety of means to generated the user-level pairings, including login information.

	Device ($\gamma = 0.0$)	User ($\gamma = -6.0$)
ID pairs with one or more feature pairs in training data	1,922,784,915	2,889,346,629
pairs with $S(i, i') > \gamma$	1,283,027,741	931, 455,977
pairs with $S(i, i') \leq \gamma$	639,757,174	1,957,890,652

Table 1: Summary of NBSS statistics. Counts are the number of pairs, $i \neq i'$.

	Device	User
scored ID pairs	1,922,784,915	2,889,346,629
distinct IDs	619,607,578	667,494,232
identified groups	322,006,060	273,795,191
average IDs per group	1.92	2.44

Table 2: Statistics on ID pairs and identified groupings resulting from the community detection algorithm.

Number of IDs per device/user	Count	
	Devices ($\gamma = 0.0$)	Users ($\gamma = -6.0$)
1	164,906,331	150,383,873
2	80,839,391	41,647,927
3-4	60,676,942	42,417,496
5-8	14,845,135	29,430,469
> 8	738,261	9,915,426

Table 3: Distribution of the number of IDs per user or device.

3.2 Results

We applied NBSS to all pairs of IDs that shared a cohort in the input dataset. Of these 3.1 billion pairs under test, in the case of device-level learning, 1.9 billion ID pairs had at least one pair of features that existed in the set of *training pairs*, and were scored by the NBSS approach. For user-level groupings, 2.9 billion pairs under test had one or more feature pairs in the training set. Recall that a large score provides evidence that the IDs belong to the same device/user, while a small (negative) score indicates the IDs belong to different devices/users. Table 1 summarizes the results.

Next, at the device level, the 1.9 billion pairs of IDs scored by NBSS were clustered using a variant of Louvain Modularity (see Section 2.4). The 1.9 billion pairs contained 620 million distinct IDs, which were grouped into 322 million small groups corresponding to devices. Each small group was assigned a group ID, *i.e.,* a *device ID*. Table 2 shows summary statistics on ID pairs and identified groupings resulting from the community detection algorithm. Table 3 shows a distribution of the number of IDs per group.

At the user level, the 2.9 billion pairs of IDs consisting of 667 million unique IDs were grouped into 274 million small groups corresponding to users. These groups average 2.44 IDs, with the distribution of group sizes given in Table 3.

3.3 Validation

Validation of the scores produced by the NBSS approach and the set of groupings produced by the community detection algorithm was completed using *leave-one-out cross-validation.* Leave-one-out cross-validation individually tests each labeled example by withholding that example from the training set.

Scores produced by the NBSS algorithm were verified as follows. As noted, the client panel was used to create 336,087 pairs, each with a label indicating if the pair of IDs belong to the same MAC address, or a different MAC address. Each pair of IDs belong to one of 13,068 households. The pairs were then scored using the other 13,067 households. In particular, the quantities $N_1(x_\ell, x'_m)$ and $N_0(x_\ell, x'_m)$ in equation (2) are the count of example pairs with feature pair x_ℓ, x'_m, excluding the pairs sharing the same CP household identifier as the pair under test. If exclusion of data from the household under test resulted in $N_1(x_\ell, x'_m) + N_0(x_\ell, x'_m) \leq 2$, then the feature pair x_ℓ, x'_m was not used in holdout scoring.

At the device level, the resulting dataset was comprised of 199,318 ID pairs, an associated *leave-one-out* NBSS score defined by (1), and a label indicating if the pair of IDs belonged to the same device. We note this implies that a significant number (136,769) of pairs had no examples in training data after holdout of the household under test, and were thus not scored by the algorithm. We investigated assigning unscored pairs a score using an unsupervised approach, but we omit those results for brevity.

The same process of exclusion of a household under test was followed for user-level scoring. A much smaller fraction of ID pairs had no examples in training data after holdout of the household under test, largely due to the significantly larger scale of the training data.

In both cases, the score was compared against a threshold, denoted γ, to derive the ROC curves shown in Fig. 3 and Fig. 4. A true positive is declared if $S(i, i') > \gamma$ and ground truth indicates IDs i and i' belong to the same device/user. Likewise, a true negative is declared if $S(i, i') \leq \gamma$ and ground truth indicates IDs i and i' belong to different devices/users. $S(i, i')$ is defined in (1) for NBSS and Sec. 2.3 for trigram, intersection, and the IP-colocation score. Two traces are shown for the NBSS approach: one restricted to the subset of pairs scored by the algorithm (*i.e.,* the 199,318 pairs with one or more examples in training data), and one which includes all pairs under test (*i.e.,* all 336,087 pairs). In the latter case, any pair that was not scored by the algorithm was assigned a score of zero.

The results show, perhaps not surprisingly, that NBSS outperforms the other approaches. The unsupervised approach based on trigrams performs second best. In the case of user-level groupings, inside a cohort, the IP-colocation score is negatively correlated with devices belonging to the same user and perform worse than random score assignment. We postulate this may be a result of cookie resetting behavior or bias in training data.

We also note that the true positive and false positive rates are better for device-level groupings than user-level groupings. This reflects the underlying difficulty of the problem: users have a variety of usage behaviors that ultimately might be less predictive of user/ID association. Conversely, hardware characteristics such as the UA and screen size are very much indicative of the relationships between device/ID association.

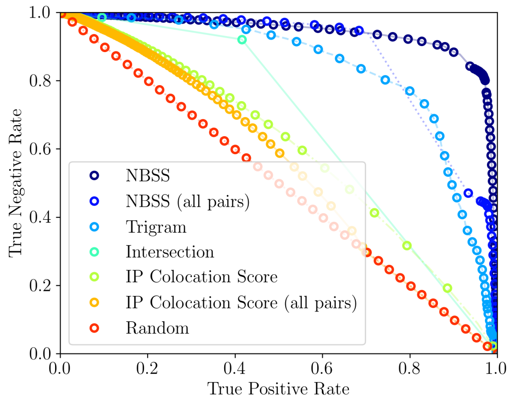

Figure 3: *Device-level* receiver operating characteristic (ROC) curve showing true positive and true negative rates of pairs under test using leave-one-out cross-validation.

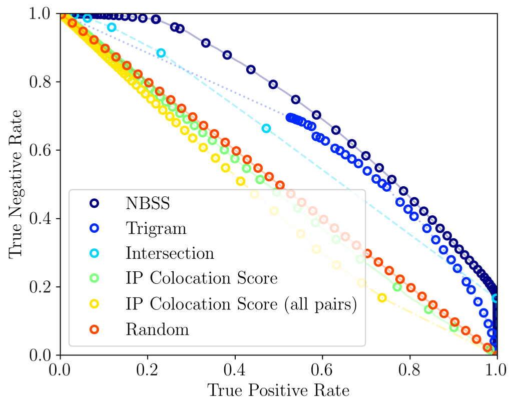

Figure 4: *User-level* ROC curve showing true positive and true negative rates of pairs under test using leave-one-out cross-validation.

For each scoring approach highlighted above, community detection as described in Section 2.4 was applied to the 336,087 ID pairs at the device level. Unscored pairs were assigned a score of zero. The same process was followed at the user level.

Precision and recall analysis was implemented as follows. A ground truth device was defined as a group of IDs that share a common MAC address in the training set. A ground truth user was defined as a group of IDs that share a common user ID in the user-level training set. Application of the community detection algorithm to scored pairs return a potentially different group of IDs that approximate the ground truth device/user. For each ground truth group, the smallest test group with maximal intersection of IDs was used to compute precision and recall. Precision is defined as the number of IDs the two groups have in common, divided by the number of IDs in the produced group. Recall is the number of device IDs the two groups have in common, divided by the number of IDs that belong to the ground truth group. The mean precision and recall values were computed across all ground truth devices and users.

By applying a universal additive scalar to the similarity scores prior to implementing community detection, a precision/recall curve can be generated. The additive scalar serves as a tuning parameter, and can be interpreted as adjusting the prior probability of the classes. Ultimately, the additive scalar allows gross adjustment of the size of the groupings output by the community detection algorithm.

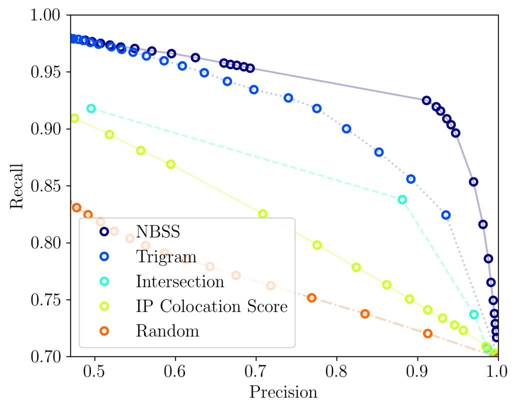

Figure 5: *Device-level* precision recall curve derived using leave-one-out cross-validation.

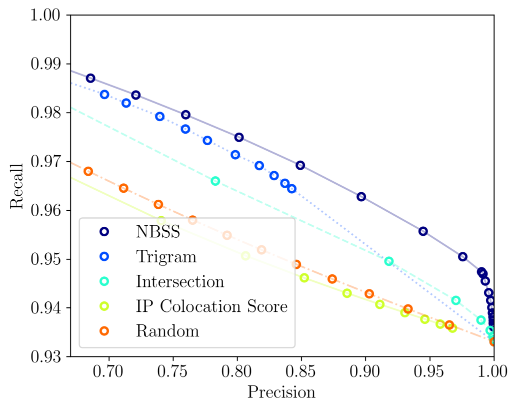

Figure 6: *User-level* precision recall curve derived leave-one-out cross-validation.

The precision and recall results using leave-one-out cross-validation for device-level community detection are shown in Fig. 5. For device-level groupings, mean precision and recall exceed 91%. Notably, the lower right most point on the plots shows the mean precision and recall when all IDs are grouped by themselves. This highlights the approximately ten fold bias in the training data towards pairs of IDs that do not belong to the same users - at the user level, 93% recall can be achieved by grouping IDs alone. Lastly, we note that the validation approach does not test the quality of the cohorts used as input. Both the ROC curve and the precision/recall plots only showcase the performance of our approach for grouping users/devices presented here.

Positive predictive value (or accuracy) is the fraction of identified positives that are true positives. A plot of accuracy vs. scale can be established by plotting accuracy from the leave-one-out cross-validation against the number of pairs identified as positive in the test set. Results indicate scale of 500 million pairs with an accuracy of 85% for device-level groupings (Fig. 7), and scale of 300 million pairs with an accuracy of 70% for user-level groupings (Fig. 8).

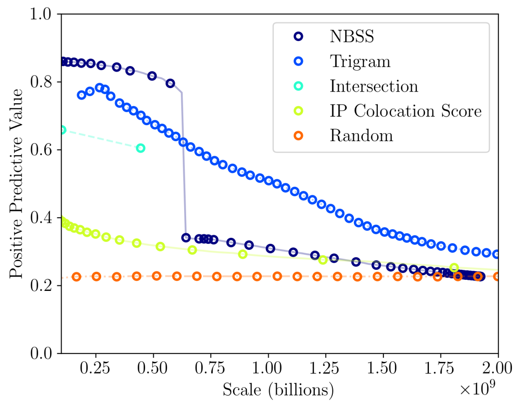

Figure 7: *Device-level* accuracy vs. scale. Accuracy is determined by using leave-one-out cross-validation, while scale is derived from counts of total pairs in the test set.

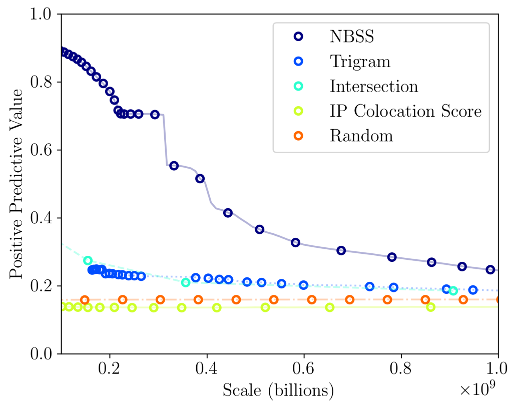

Figure 8: *User-level* accuracy vs. scale. Accuracy is determined by using leave-one-out cross-validation, while scale is derived from counts of total pairs in the test set.

Our results from validation and testing can be summarized as follows: Using leave-one-out cross-validation on ground truth data provided by a client panel, we find that our methodology is able to associate sets of IDs with individual devices with high accuracy. We find that the accuracy on user-level assignments is less accurate.

4 IMPLEMENTING DEVICE/USER IDENTIFICATION AT SCALE

Our objectives in implementing our methodology were to create an efficient, extensible and quality-assured processing pipeline that could be run on a daily basis at a scale of billions of measurements. To accomplish this, we implemented our methodology in Pig [33], an open source parallel processing language maintained by the Apache Foundation that is designed to run on Hadoop clusters [1]. The total codebase is several thousand lines of code, roughly half of which is devoted to data integrity checks on input and output streams. We take advantage of the substantial overlap between device- and user-level classification taks, sharing code when possible. Input to the implemented pipelines are aggregates of the feature bags described in Sections 3.1.1 and 3.1.2, which are processed daily. Training and validation tasks are performed on a weekly basis, resulting in device- and user-level groupings that inform downstream pipelines.

5 PRIVACY

Over the past decade, there have been growing concerns over user privacy related to IDs. Many articles on this subject have been published in the popular press including calls for government regulation on tracking (*e.g.,* [16, 23, 30]). Indeed, it is illegal to track users via cookies and advertising IDs in the EU without explicit consent [34] and the impending General Data Protection Regulation in the EU will have an impact on third-party tracking [3]. These privacy concerns have sparked the development of tools that facilitate cookie management and removal (*e.g.,* [29]). Finally, a number of studies have focused on understanding the prevalence and characteristics of tracking via IDs and the related issue of information leakage (*e.g.,* [11, 21, 22, 27]). We are informed by all of these studies and are sensitive to privacy concerns. However, our work differs from prior studies in its specific focus on the problem of relating multiple IDs to a single device or user.

While steps to improve privacy and reduce unwanted tracking on the internet are laudable, the issue is complex. Incomplete and inaccurate third-party device graphs result in inefficiencies that propagate through the digital advertising ecosystem. Features such as iOS 11's Intelligent Tracking Prevention further tip the scale in favor of entities with access to large first-party deterministic graphs based on login information (i.e, Facebook, Google, Apple, and others). Third-party measurement and validation entities, such as comScore, face increased difficulty in providing independent and accuracte tools for validation of delivery of digital advertisements and measurement of audience size. Advertisers not privy such graphs estimate the resulting loss of ad revenue into the hundreds of millions of dollars [6]. Regardless of perspective, third-party device graphs will continue to play a role in online advertising. Making accurate probabilistic graphs in spite of tracking prevention and the multiplicity of IDs is a challenging and timely problem. We posit that by describing techniques for making these associations, we can expand the conversation about how best to describe and implement privacy policies to protect the users.

6 RELATED WORK

While there is limited academic literature using the taxonomy 'internet device graphs' beyond [24], the problem of identification in the internet has a rich history. The value of persistent identification has led to development of a variety of techniques that go beyond standard cookies and advertising IDs. An example is the Evercookie, which, as the name suggests, is cookie that cannot be easily cleared by a client [19]. Evercookies are enabled by storing cookie data in a number of different client-accessible locations. Adobe and Microsoft created similar functionality, which has since been discontinued due to security and privacy concerns [13, 26].

Another example of an alternative to cookies and advertising IDs are *device fingerprinting* techniques that use features readily accessible for unique association at both the browser level [14, 28] and device level (also termed cross-browser identification) [9, 12]. Cross-browser identification has relied on direct association of observed features, either IP address-based [9] or hardware and OS features exposed by some web protocols [12], as opposed to learning-based methods. These techniques have been shown to have varying levels of accuracy [32] and there are a number of online services that will test the "trackability" of a user's browser (*e.g.,* [15]). In contrast to the device fingerprinting techniques, the signals we use (*i.e.,* the features) are extremely limited and easily gathered in web measurements. This is both to demonstrate the feasibility of our method without complex signals, and for practicality. A large feature space dilutes the power of the training set. Nonetheless, many of the features suggested in the fingerprinting literature could be included in implementations of the methodology presented here.

Our method for determining whether two IDs are related falls into a larger class of *Bayesian learing*, *similarity learning*, and *distance metric learning* methods [7, 37]. Bayesian learning and naïve Bayes has a rich history dating back more than 50 years [25]. The naïve Bayesian similarity learning approach presented here can be seen as a straightforward extension of standard naïve Bayes, where one takes a cross product between the individual feature sets to create a single features set.

Much of the work in distance metric learning focuses on regression-based methods. Yang *et al.* describe a Bayesian framework for *active* distance metric learning that bears some similarity [38]. Other methods are used for measuring the similarity between two objects to organize them into disjoint clusters [39]. In [17], the authors propose new algorithms to compute the similarity between sets of overlapping clusters. In [36], the authors develop a similarity metric and show success on text document clustering. Nonetheless, our formulation and application are unique.

Finally, as discussed in Section 2, our work assumes that a coarse association of IDs in cohorts has already been made. Recent work by Malloy *et al.* describes a method for generating a basic *device graph* from internet measurements [24]. Commercial device graph offerings are widely available *e.g.,* [2, 5, 35]. The method in [24] utilizes IP-colocation *i.e.,* the observation of two or more IDs on a given IP address, to establish relations between IDs. It then describes a community detection method for creating household groupings. While this method is useful in our work, we only require that some kind of coarse association is made – households cohorts are not required. With a perspective focused on application, this work can be viewed as a logical extention and improvement to [24] that produces fine-grained device and user groupings of IDs.

7 SUMMARY AND FUTURE WORK

In this paper, we investigate the problem of associating multiple IDs (*i.e.,* web cookies and advertising IDs) with a single device or user. This problem is important for entities that track user behavior (both first- and third-parties) and in understanding related privacy implications. We develop a methodology that can be implemented after IDs are grouped into small coarse groups called cohorts. The methodology follows two steps. First, we apply a Bayesian similarity algorithm that uses examples of pairs of devices and their associated telemetry, including the IP address and UA. After pair-wise scores between device identifiers are derived, the second step is to apply a variant of the Louvain Modularity community detection to group identifiers that belong to the same device or user. We trained and assessed our method using a unique dataset collected from a client panel. Results from leave-one-out cross-validation show sensitivity and specificity rates of 90% for device-level groupings and 64% for user-level groupings, indicating the relative difficulty of the task at the user level. Next, we apply our methodology to a dataset consisting of 700 million device identifiers collected over the course of six weeks in the United States. We train and validate our methodology using a unique dataset collected from a client panel with full visibility. Results from leave-one-out cross-validation show mean precision and recall of 91% and 92% for association of identifiers at the device level, and precision and recall of 94% and 95% at the user level. In total, processing of the training, validation, and evaluation phases for the roughly 700 million IDs complete in hours in a large Hadoop cluster. The pipeline remains very general - all that must be provided is an input partitioning of IDs and a method of extracting features for a given ID.

REFERENCES

[1] 2017. Apache Pig. https://pig.apache.org/. (2017).
[2] 2017. Drawbridge. http://www.drawbridge.com/c/graph/. (October 2017).
[3] 2017. The EU General Data protection Regulation. (October 2017). http://www.eugdpr.org/
[4] 2017. Intelligent Tracking Prevention. https://webkit.org/blog/7675/intelligent-tracking-prevention/. (October 2017).
[5] 2017. Lotame Cross Device. http://www.lotamecrossdevice.com. (October 2017).
[6] 2018. Apple Tracking Block Costs Advertising Companies Millions. *The Guardian* (January 2018).
[7] A. Bellet, A. Habrard, and M. Sebban. 2015. *Metric Learning.* Morgan and Claypool.
[8] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), P10008.
[9] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. 2011. User tracking on the web via cross-browser fingerprinting. In *Nordic Conference on Secure IT Systems.* Springer, 31–46.
[10] Aaron Cahn, Scott Alfeld, Paul Barford, and S Muthukrishnan. 2016. An Empirical Study of Web Cookies. In *Proceedings of the 25th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 891–901.
[11] A. Cahn, S. Alfeld, P. Barford, and S. Muthukrishnan. 2016. An Empirical Study of Web Cookies. In *Proceedings of the World Wide Web Conference (WWW '16).* Montreal, Canada.
[12] SL Yinzhi Cao and E Wijmans. 2017. Browser Fingerprinting via OS and Hardware Level Features. In *Proceedings of the 2017 Network & Distributed System Security Symposium, NDSS*, Vol. 17.
[13] Electronic Privacy Information Center. 2017. Local Shared Objects – Flash Cookies. (October 2017). https://epic.org/privacy/cookies/flash.html
[14] P. Eckersley. 2010. How Unique is Your Web Browser?. In *In Proceedings of the International Symposium on Privacy Enhancing Technologies.* Berlin, Germany.

[15] Electronic Frontier Foundation. 2017. Panopticlick. (October 2017). https://panopticlick.eff.org/

[16] V. Goel. 2014. California Urges Websites to Disclose Online Tracking. *The New York Times* (May 2014).

[17] Mark K. Goldberg, Mykola Hayvanovych, and Malik Magdon-Ismail. 2010. Measuring Similarity Between Sets of Overlapping Clusters. In *Proceedings of the 2010 IEEE Second International Conference on Social Computing (SOCIALCOM '10)*. IEEE Computer Society, Washington, DC, USA, 303–308.

[18] Sergio Gomez, Pablo Jensen, and Alex Arenas. 2009. Analysis of community structure in networks of correlated data. *Physical review. E, Statistical, nonlinear, and soft matter physics* 80, 1 (July 2009), 016114.

[19] S. Kamkar. 2017. Evercookie. (October 2017). https://github.com/samyk/evercookie/commits/master

[20] Jeffery Kline, Aaron Cahn, Paul Barford, and J Sommers. 2017. n the Structure and Characteristics of User Agent Strings. In *Proceedings of the ACM Internet Measurement Conference, IMC*.

[21] B. Krishnamurthy, D. Malandrino, and C. Wills. 2007. Measuring Privacy Loss and the Impact of Privacy Protection in Web Browsing. In *Proceedings of the Symposium on Usable Privacy and Security*. Pittsburgh, PA.

[22] B. Krishnamurthy and C. Wills. 2006. Generating a Privacy Generating a Privacy Footprint on the Internet. In *Proceedings of the ACM Proceedings of the Internet Measurement Conference*. Rio de Janerio, Brazil.

[23] E. Lee. 2011. Sen. Rockefeller: Get Ready for a Real Do-Not-Track Bill for Online Advertising. *AdAge* (May 2011).

[24] Matthew Malloy, Paul Barford, Enis Ceyhun Alp, Jonathan Koller, and Adria Jewell. 2017. Internet Device Graphs. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1913–1921.

[25] Melvin Earl Maron and John L Kuhns. 1960. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)* 7, 3 (1960), 216–244.

[26] J. Mayer. 2011. Tracking the Trackers: Microsoft Advertising. (August 2011). http://cyberlaw.stanford.edu/blog/2011/08/tracking-trackers-microsoft-advertising

[27] J. Mayer and J. Mitchell. 2012. Third-Party Web Tracking: Policy and Technology. In *Proceedings of the IEEE Symposium on Security and Privacy*. San Francisco, CA.

[28] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham. 2011. Fingerprinting Information in JavaScript implementations. In *Proceedings of Web 2.0 Security and Privacy Workshop (W2SP)*. Oakland, CA.

[29] Mozilla. 2017. BetterPrivacy. (2017). https://addons.mozilla.org/en-US/firefox/addon/betterprivacy/

[30] M. Murgia and D. Robinson. 2016. Google faces EU curbs on how it tracks users to drive adverts. *The Financial Times* (December 2016).

[31] M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E* 69 (Feb 2004), 026113. Issue 2. https://doi.org/10.1103/PhysRevE.69.026113

[32] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. 2013. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In *In Proceeding of the IEEE Symposium on Security and Privacy*. San Francisco, CA.

[33] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. 2008. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 1099–1110.

[34] Optanon. 2017. The Cookie Law Explained. (October 2017). https://www.cookielaw.org

[35] TAPAD. 2017. The TAPAD Device Graph. (October 2017). https://www.tapad.com/device-graph/

[36] G. Torres, R. Basnet, A. Sung, S. Mukkamala, and B. Ribiero. 2008. A Similarity Measure for Clustering and its Applications. *Proceedings of World Academy of Science, Engineering and Technology* 31 (2008), 490–496.

[37] L. Yang and R. Jin. 2017. Distance metric learning: A comprehensive survey. (October 2017). https://www.cs.cmu.edu/~liuy/frame_survey_v2.pdf

[38] L. Yang, R. Jin, and R. Sukthankar. 2007. Bayesian Active Distance Metric Learning. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*. Vancouver, Canada.

[39] Ying Zhao and George Karypis. 2005. Data clustering in life sciences. *Molecular Biotechnology* 31, 1 (2005), 55–80.