Clustering for Private Interest-based Advertising

Alessandro Epasto[*]
aepasto@google.com
Google, USA

Andrés Muñoz Medina[*]
ammedina@google.com
Google, USA

Steven Avery
sgavery@google.com
Google, USA

Yijian Bai
ybai@google.com
Google, USA

Robert Busa-Fekete
busarobi@google.com
Google, USA

CJ Carey
cjcarey@google.com
Google, USA

Ya Gao
rosygao@google.com
Google, USA

David Guthrie
dguthrie@google.com
Google, USA

Subham Ghosh
subhamghosh@google.com
Google, USA

James Ioannidis
ioannidis@google.com
Google, USA

Junyi Jiao
junyijiao@google.com
Google, USA

Jakub Lacki
jlacki@google.com
Google, USA

Jason Lee
jdlee@google.com
Google, USA

Arne Mauser
amauser@google.com
Google, USA

Brian Milch
brian@google.com
Google, USA

Vahab Mirrokni
mirrokni@google.com
Google, USA

Deepak Ravichandran
deepakr@google.com
Google, USA

Wei Shi
shiw@google.com
Google, USA

Max Spero
maxspero@google.com
Google, USA

Yunting Sun
ytsun@google.com
Google, USA

Umar Syed
usyed@google.com
Google, USA

Sergei Vassilvitskii
sergeiv@google.com
Google, USA

Shuo Wang
mapleisle@google.com
Google, USA

ABSTRACT

We study the problem of designing privacy-enhanced solutions for interest-based advertisement (IBA). IBA is a key component of the online ads ecosystem and provides a better ad experience to users. Indeed, IBA enables advertisers to show users impressions that are relevant to them. Nevertheless, the current way ad tech companies achieve this is by building detailed interest profiles for individual users. In this work we ask whether such fine grained personalization is required, and present mechanisms that achieve competitive performance while giving privacy guarantees to the end users. More precisely we present the first detailed exploration of how to implement Chrome's Federated Learning of Cohorts (FLoC) API. We define the privacy properties required for the API and evaluate multiple hashing and clustering algorithms discussing the trade-offs between utility, privacy, and ease of implementation.

CCS CONCEPTS

• **Theory of computation** → **Unsupervised learning and clustering**; • **Information systems** → **Online advertising**; • **Security and privacy** → **Privacy protections**.

KEYWORDS

Interest-based advertising; clustering; anonymity; privacy

ACM Reference Format:
Alessandro Epasto, Andrés Muñoz Medina, Steven Avery, Yijian Bai, Robert Busa-Fekete, CJ Carey, Ya Gao, David Guthrie, Subham Ghosh, James Ioannidis, Junyi Jiao, Jakub Lacki, Jason Lee, Arne Mauser, Brian Milch, Vahab Mirrokni, Deepak Ravichandran, Wei Shi, Max Spero, Yunting Sun, Umar Syed, Sergei Vassilvitskii, and Shuo Wang. 2021. Clustering for Private Interest-based Advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August*

[*]Both authors contributed equally to this research.

This work is licensed under a Creative Commons Attribution International 4.0 License.

KDD '21, August 14–18, 2021, Virtual Event, Singapore.
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8332-5/21/08.
https://doi.org/10.1145/3447548.3467180

14–18, 2021, Virtual Event, Singapore. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467180

1 INTRODUCTION

Interest based advertising (IBA) is a key part of the modern Internet advertising ecosystem, as it allows advertisers to show users ads based on their interests. IBA lies in contrast to contextual based advertising, where the impression shown depends on the content of the website the user is browsing. Thus IBA allows a car advertiser to advertise to people who are in-market for a new car, no matter which site they are browsing, whereas contextual advertising would take cues from the site's content to decide whether a car ad is appropriate.

This ability to reach people based on their latent interests is a powerful primitive that raises the efficiency of the market: users see more relevant ads, advertisers see better performance, and publishers garner higher revenues. However, to enable IBA today, ad tech companies build detailed interest profiles for individual users. As of 2021 there are 4.66 billion internet active users [44] and by some estimates [38], the information collected by 52 advertising companies can recover, on average, 91% of a user's browsing history. In this work we ask whether such fine grained personalization is required, and present mechanisms that achieve competitive performance while giving privacy guarantees to the end users.

Specifically, we focus on the role that *third-party cookies* play in building user profiles, and, following recent announcements by Safari, Firefox, and Chrome, explore whether IBA can be enabled without their use. Our starting point is the *Federated Learning of Cohorts* (FLoC) API proposed by Chrome as part of the Privacy Sandbox initiative. Importantly, the FLoC API can act as a drop-in *privacy-safe* replacement for third-party cookies. This is in stark contrast to other solutions for private online advertising [23, 29] that would require big changes from ad tech companies to conform to a new advertising paradigm. We note that in practice such major changes run the risk of ad-techs turning to fingerprinting and other nefarious methods to preserve the status quo, rather than adopting the new APIs.

At a high level, the approach taken by FLoC is to group users into k-anonymous groups, and allow profile building per group, rather than per individual. While the idea is relatively simple, it is not a priori clear that such a grouping can be both efficiently computed in a private manner, and be effective enough to replace current per-user profiles. In this work we:

- Formally define the FLoC clustering problem, setting specific privacy and utility goals.
- Introduce a number of different algorithms that present a Pareto frontier between privacy, utility, and complexity of implementation.
- Evaluate these methods on public and private datasets, demonstrating the trade-offs involved, and proving the viability of an IBA system based solely on cohorts.

Our paper is organized as follows: we first discuss the extensive literature on private advertising. We then describe the current state of IBA and the privacy risks involved with building personalized interest profiles. After this, we introduce the FLoC API and discuss its technical requirements and constraints. The rest of the paper is devoted to casting the problem of designing the FLoC API as a clustering problem with cluster size constraints. In Section 4 we discuss several candidates for the clustering algorithm each with different utility and privacy trade-offs. Section 5 contains an extensive evaluation of the aforementioned algorithms on two public datasets: the MovieLens dataset [26] and the "Million Song Dataset" [11]. We conclude the paper with experiments on a proprietary ads dataset where we demonstrate that using the FLoC API can provide comparable utility to the current cookie-based system while giving stronger privacy protections to end users.

2 RELATED WORK

Our work spans the areas of privacy preserving technology in advertising as well as the algorithmic problem of clustering. These two areas are vast so we will only briefly review the most relevant material.

Privacy in advertisement. The privacy issues associated with online advertising are very well known in the research community. A thorough review of these problems can be found in [19]. Multiple solutions on how to solve this problem have been proposed before. For instance, Guha et al. and Toubiana et al. [23, 29, 45] propose systems based on cryptographic components to preserve user privacy and interest-based advertising. These works propose a completely new online advertising framework. Unlike the FLoC API, this framework requires all ad tech players to modify their internal systems and implementations. Moreover, their work does not discuss in detail the effects of their proposed systems on the quality of ad targeting. By contrast, we propose a detailed analysis of the effects of privacy on the ability to build interest profiles.

A strong way of preventing cross-site tracking is given by Kazienko [32]. They propose targeting users based only on their behavior across a *session*. These sessions are domain specific, effectively reducing this proposal to contextual advertising. A different approach to user privacy is the one proposed by AdNauseum [27]. Here the authors generate artificial user profiles by blocking ads and artificially clicking on them. By doing this, AdNasueum introduces enough noise in a user profile that it removes any incentive for ad tech companies to track a user. This approach, however, does not actually stop the tracking of users nor does it take into account how publishers get affected by blocking ads or by generating fake profiles.

Clustering. Clustering is the quintessential unsupervised machine learning problem and we refer to [46] for a review of this area. The study of clustering dates back more than 5 decades [22, 28], and countless applications have been fond for clustering methods including data summarization and anonymization, exploratory data analysis, matrix approximations and outlier detection as well as the study of social and biological networks [15, 28, 34, 37, 40, 42, 47]. Given the vast array of applications for clustering, it is no surprise that multiple independent formulations of it have been developed. This includes k-clustering in metric spaces [2, 21, 36], hierarchical clustering [9, 13, 16, 39], and graph clustering and community detection [20], among others. We will briefly review these directions.

Perhaps the most well-known clustering formulation is that of k clustering in metric spaces, where one seeks k points (known as centers) from the input so as to minimize an objective depending on the distances of all points in input to their closest center. Several variants of this problem have been studied including k-median, k-means, and k-center. All such problems are NP-hard, but many constant factor approximation algorithms are known [2, 21, 36]. This area has generated numerous research directions including the study of core-sets [8, 25], distributed algorithms [6], and streaming methods [3]. Many other directions in metric clustering (not center based) have been explored including density based clustering [18].

Related to our work is also the area of size-constrained clustering where one seeks clusters with size larger than a specified constant. This is often the case in many applications in which clustering is used for anonymization [1, 12] or when privacy considerations require all the clusters to satisfy a minimum size guarantee as in our work. Perhaps one the most relevant formulation of the problem to our paper is that of r-gather [5] where the only requirement is that clusters have size at least r but the number of clusters is not constrained. For this problem, concurrent work [17] studied efficient distributed algorithms that can scale to large data scenarios typical in the application addressed in this work.

All previously mentioned formulations concern flat clustering (a single partitioning of the data) but an important challenge in the area is that of hierarchical clustering which seeks a hierarchical family of clusters representing the data at different granularities [22, 39]. Such formulation has many advantages over flat clustering including avoiding the need to specify a precise number of clusters (which is often unknown). Significant work in this area has studied procedures for hierarchical clustering including the well-known single linkage, complete linkage and average linkage [16, 22] as well as efficient distributed hierarchical algorithms [9].

Another relevant area is that of graph clustering and community detection [20, 35]. Work in these areas has focused on finding tightly-knit clusters of nodes in a graph which can represent, for instance, fundamental organizational units in social networks as well as in biological networks. Graph clustering and community detection have been defined in terms of spectral methods optimizing conductance [4], density based methods [33] and correlation clustering [7] among others.

3 INTEREST BASED ADVERTISING

We will now describe Interest Based Advertising (IBA) and how third-party cookies are used to enable this technology. We begin by defining the main concepts.

- A *user* is any person browsing online.
- A *publisher* is the landing website which a user has the intent to visit. This is usually referred to as a first-party domain. An example of a publisher is `nytimes.com`. An advertiser website, for instance `nike.com`, can also act as a publisher when a user visits that website directly.
- An *advertiser* is an entity who wants to show an ad to a user. Ads are shown on publisher's sites.
- *Ad tech companies* are intermediaries working on behalf of publishers—sell side platforms (SSPs), advertisers—demand side platforms (DSPs), or both. Ad tech companies are responsible for the technical and algorithmic aspects of fulfilling ad requests. Ad tech companies are usually referred to as third-parties.
- A *cookie* is a byte string associated with a (user, domain) pair which is stored in a user's browser. A cookie corresponds to the identity of a user in a domain. A cookie is called first-party when it corresponds to the domain the user is currently visiting and third-party otherwise. Cookies are crucial to enable most features in modern web browsers as they allow publishers to preserve state. For instance, keeping a shopping cart up to date or keeping a user logged in.

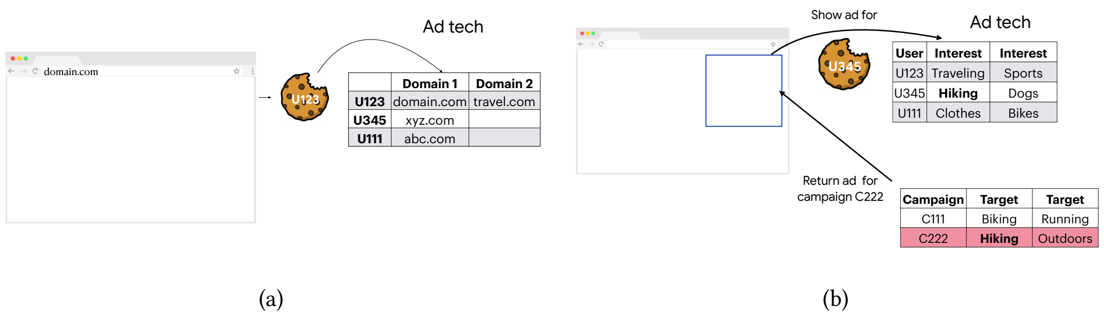
The role of third-party cookies. We now describe how third-party cookies are currently used to enable interest based advertising. When a user visits a website (for instance domain.com), the website sends an HTTP request to an ad tech company with the (third-party) cookie the ad tech company has for this user. After visits to multiple websites, the ad-tech company can build a *profile* of the websites visited by the user (See Figure 1(a)). The ad tech company uses this profile to infer user interests, and at a future time, when the ad tech company receives an ad request, the ad tech company can look up the user profile and match it with their own targeting campaigns (see Figure 1(b)).

The above discussion also describes the main user privacy concern. Should ad tech companies be allowed to store a detailed browsing history of users? The main information needed by an ad tech company to serve personalized ads are user profiles, and the identity and browsing history of a user are currently used only as means to an end. Therefore, it should be possible for personalized ads to be served without having to keep a detailed record of every user's browsing history.

3.1 The FLoC API

Recently Chrome has proposed [30] the Federated Learning of Cohorts (FLoC) API as a way to enable IBA in a world without third-party cookies. The goal of this API is to replace these unique user identifiers by a shared, k-anonymous [41] cohort id. Just like third-party cookies, a cohort id might have no meaning on its own. However, this cohort id can be used by ad tech companies to build *cohort profiles.* These profiles can then be used as before to match ad campaigns. Since a cohort id is shared by multiple users, when used on their own, a cohort id does not allow for re-identification of users across websites. On the other hand, the profiles built by the ad tech company are now interests shared by a large number of users, which in principle could hinder the ability to accurately target users. Ideally, cohorts would consist of a large number of users who share exactly the same interests. This naturally raises the question of how should cohort ids be assigned to users? While this can be naturally interpreted as a clustering problem - *similar* users get clustered together, we need to keep in mind the following constraints.

- **K-anonymity** [41]. Each cohort id must be shared by at least k users. By enforcing k-anonymity we ensure that a user is *lost in the crowd.*
- **Local computation**. The cohort id needs to be computed in the browser, preferably in a way that can be easily audited.
- **Central server trust** State-of-the-art clustering algorithms generally require access to raw data, which in this case would



	Domain 1	Domain 2
U123	domain.com	travel.com
U345	xyz.com	
U111	abc.com	

User	Interest	Interest
U123	Traveling	Sports
U345	**Hiking**	Dogs
U111	Clothes	Bikes

Campaign	Target	Target
C111	Biking	Running
C222	**Hiking**	Outdoors

Figure 1: (a) Depiction of how third-party cookies are used by ad-tech companies to build user profiles. (b) Use of third-party cookies to serve personalized ads.

require access to all users' browsing history. An ideal clustering algorithm would forego this requirement, eschewing the need for any central server to collect or keep detailed browsing history of users. It is worth pointing out, however, that a single trusted server having access to this data is still better than the current state of IBA, where multiple unregulated companies have access to this fine-grained user information. In our algorithm comparison we include both centralized and distributed algorithms.

4 ALGORITHMS

We now describe the algorithms that satisfy the desiderata from the previous section, we present them here in order of simplicity as we believe ease of implementation, interpretability and debugging should be considered in the design of a clustering algorithm for the FLoC API. Throughout the remainder of the paper we will encode users as a d dimensional vectors $x_1, \ldots, x_n \in \mathbb{R}^d$ and we will denote by $\mathcal{U}$ this set of n users. A clustering algorithm is a function $F\colon \mathbb{R}^d \to \mathbb{N}$ that maps a user to a cohort id. Given a cohort id $j \in \mathbb{N}$ we denote by $C_j = \{x \in \mathcal{U} \colon F(x) = j\}$. We will drop the dependence on the cohort id j when it is understood from context. Finally, given a cohort C we let

$$\mu(C) = \frac{1}{|C|} \sum_{x \in C} x$$

denote the *centroid* of a cohort.

4.1 SimHash

SimHash is an instantiation of the popular locality sensitive hashing (LSH) family of algorithms [48]. Initially developed with the goal of identifying near duplicate documents quickly, SimHash takes as input a d-dimensional vector x and outputs a p-bit vector $H_p(x) \in \{0,1\}^p$ which we refer to as the hash of x.

The i-th coordinate of the hash vector is obtained by the following rule:

$$H_p(x)[i] = \begin{cases} 0 & w_i^\top x \le 0 \\ 1 & w_i^\top x > 0, \end{cases}$$

where $w_1, \ldots, w_p$ are random unit-norm vectors. A cohort corresponds to all users whose input vectors share the same hash. Figure 2 shows an example of the SimHash function H_3.

SimHash has the property that similar vectors are more likely to be hashed to the same cohort id than dissimilar vectors. More precisely, if x_1 and x_2 are two vectors, then the probability of mapping x_1 and x_2 to the same p-bit cohort id is given by:

$$\mathbb{P}(H_p(x_1) = H_p(x_2)) = \left(1 - \frac{\theta(x_1, x_2)}{\pi}\right)^p,$$

where $\theta(x_1, x_2)$, corresponds to the angle between vectors x_1 and x_2. That is, input vectors with small angles between them are exponentially more likely to share the same hash than input vectors with a large angle between them. Alternatively, vectors with high *cosine similarity*, $\frac{x_1^\top x_2}{\|x_1\|\|x_2\|}$, are more likely to be in the same cohort.

The main advantage of using SimHash is that the computation of the cohort id for one user does not depend on the information of others. Given a vector x, its cohort id can be calculated in the client without knowledge of any other user's information. The properties of SimHash ensure that the cohort id calculated in this manner is shared with users that have similar input vectors. In particular, there is no need for any centralized data collection to compute cohort ids. Despite this, the properties of the SimHash algorithm will ensure that cohorts generated in this way will consist of similar users. This is a remarkable feature of the SimHash algorithm as it allows for clustering to happen without a central server ever storing a user's browsing history.

The main downside of SimHash is that a minimum cluster size cannot be enforced. Nonetheless, this problem can be solved by having an anonymity server that tracks the size of each cohort. This server could block the API from returning a cohort id if the cohort is not large enough. Since the server only gets to access a small bit length hash of a user's browsing history, the amount of information revealed to the server is minimal.

4.2 SortingLSH

The choice of the number of bits p defining the SimHash algorithm is crucial. If it is too low, cohorts will be large, making it more likely for dissimilar users to be part of the same cohort. On the other hand a large value of p can result in cohort ids that are shared by only a small number of users, violating the k-anonymity requirement. The difficulty in the choice of p is exacerbated by the fact that the cohorts generated by SimHash can have very heterogeneous sizes:

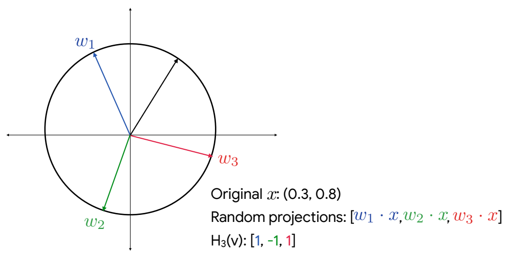

w_1
w_2
w_3
Original x: (0.3, 0.8)
Random projections: [$w_1 \cdot x$, $w_2 \cdot x$, $w_3 \cdot x$]
H3(v): [1, -1, 1]

Figure 2: Depiction of the SimHash algorithm

a few very large cohorts and a large number of small cohorts. In this scenario, *splitting* the big cohorts by increasing the number of bits for SimHash is impossible as it would break the k-anonymity of smaller cohorts (see Figures 3(a) and 3(b)).

SortingLSH is a method that solves this issue and ensures k-anonymity while improving the quality of SimHash at the same time. This is achieved by homogenizing the size of cohorts. SortingLSH is a minimally centralized method that acts by post-processing SimHash clusters to ensure k-anonymity.

Let $h_i = H_p(x_i)$ denote the p-bit hash generated by SimHash for user i. Let $\mathcal{H} = \{h_i, \dots h_n\}$ be the multiset (with repetitions) of hashes of all users. Instead of assigning cohorts by grouping together users by their SimHash, SortingLSH generates cohorts as follows:

(1) Sort $h_1, \dots, h_n$ in lexicographical order to obtain a sorted list of hashes $h_{(1)}, \dots h_{(n)}$.
(2) Assign the sorted hashes to cohorts by partitioning the order in contiguous intervals containing at least k users each.

More formally, let $a_0 = h_{(0)}, a_1, ..., a_t$ be distinct values from $\mathcal{H}$ sorted in lexicographical order. For all $j < t$ SortingLSH assigns all users whose SimHash is a string in the lexicographical order between a_j (included) and a_{j+1} (excluded) to the j-th cluster id. For $j = t$ all users whose SimHash is after a_t are in the t-th cluster. The strings $a_0, \dots a_t$ can be chosen arbitrarily from $\mathcal{H}$ subject to having at least k user in each interval. (See Figure 4 for a representation of the process.)

The ordering step ensures that contiguous hashes in this order correspond to users with mostly similar SimHash values. The size constraint of the interval ensures that the cohorts always have at least k users.

PrefixLSH. There are many options for choosing the intervals in SortingLSH as one could select the intervals by any optimization method. In our experiments, however, we use a simple approach that can be implemented at scale and which we call *PrefixLSH*. This process is similar to k-d-tree construction and the related LSH techniques of ForestLSH [10]. PrefixLSH obtains the intervals using a recursive approach. The recursion proceeds in levels starting from level 1 where we have a single interval (containing all users). At level i, we tentatively partition the current interval in two, by dividing it into two strings with i-th bit 0 (left interval) and bit 1 (right interval). If both intervals have at least k users each, we continue the recursion in both (at level $i+1$). Otherwise the current interval is final. A depiction of this process is shown in Figure 3(c). It is possible to see that this variant of SortingLSH (which is the one we use in our experiments) defines intervals in the lexicographical order that have the same bit string prefixes. The implementation of any SortingLSH method does require a central server to sort all user hashes and calculate k-anonymous cohorts. However, this information is also needed by a server that enforces k-anonymity on cohorts generated using SimHash or any other method. Therefore, the level of centralization is no worse than that of SimHash using an anonymity server.

4.3 Graph-based clustering methods

Our main focus is on designing solutions involving minimal centralization but for completeness we now turn our attention to centralized methods. In this section, we study graph-based clustering algorithms that leverage the structure of a similarity graph (i.e., a graph where similar nodes are connected) to find the clusters. More precisely, in our setting, a similarity graph is a graph where users are nodes and edges connect two users if their vectors are similar. We will focus on some example baseline solutions exploiting such techniques, evaluating efficient algorithms that are implementable in large-scale distributed computational frameworks. We stress that more algorithms could be explored, and some other options are reviewed in the related work section.

To use graph-based methods in our setting we use a three step approach. First, we build a similarity graph (this process is described in the next section). Second, we apply a graph clustering algorithm (we do this using two clustering methods: affinity [9], and METIS [31]). Finally, we post-process the clusters. We now review the three steps.

4.3.1 Step one: graph construction. The first step to use a graph-based clustering in our setting is to construct a similarity graph over the users. In our experiments, we create a user-to-user cosine-similarity weighted graph (i.e, the edge weights encode the cosine similarity of the vectors associated to the users). This is achieved by using efficient locality sensitive hashing techniques to identify pairs of users with high cosine similarity of their vector, and creating an edge only for the highest similarity pairs (the LSH method is used similarly to [24]).

4.3.2 Step two: graph clustering. Once we have a similarity graph, we can run the graph clustering algorithm. Here we evaluate two algorithms.

Affinity hierarchical clustering. We used a highly scalable graph clustering algorithm known as affinity hierarchical clustering [9]. This algorithm, in summary, performs hierarchical clustering in a bottom-up fashion, creating larger clusters by merging smaller clusters connected by highly similar edges. This is done in a way that ensures a minimum cluster size for all clusters.

More precisely, the algorithm proceeds by clustering users using hierarchical agglomerative clustering as described in [9]. However, in our algorithms we need to enforce a bound on the size of clusters, and we do so by setting a lower and upper bound on the size of clusters, abiding by these bounds while forming the clusters, and removing clusters that are too small.

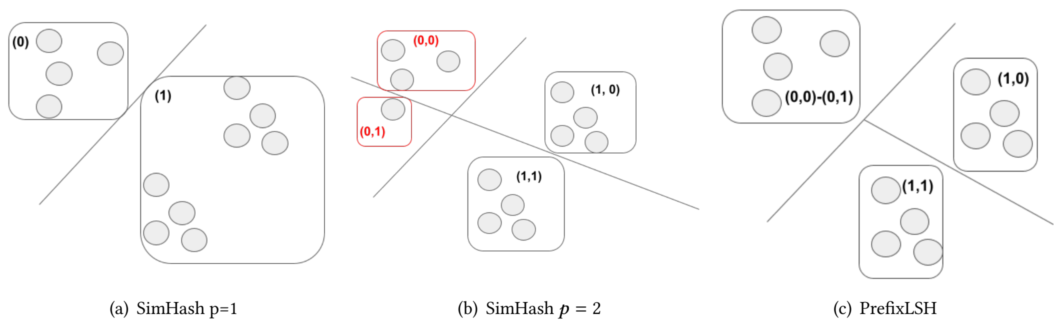


Figure 3: Example of k-anonymity enforcement for k=4. In Figure 3(a), using 1 bit of SimHash generates k-anonymous cohorts, but cohort (1) is very large. In Figure 3(b), the large cluster (1) is *split* into cohort (1,0) and cohort (1,1) by using a 2-bit SimHash. This, however, results in cohorts (0,0) and (0,1) violating the k-anonymity restriction. Finally, In Figure 3(c) only the large cluster is split while the (0,0), (0,1) is assigned to the same cohort.

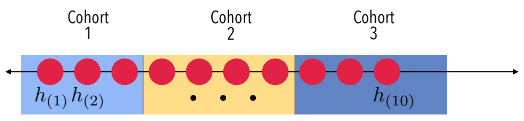


Figure 4: Example of SortingLSH enforcing k-anonymity with k=3. The first cohort consists of the first three sorted hashes, the second the 4th to the 7th and the third cohort is made up of the 8th to the 10th hash. All cohorts have at least 3 users.

METIS. As an additional baseline we use the well-known METIS graph partitioning algorithm [31]. This is a popular baseline amongst the single-machine graph partitioning algorithms with size constraints. This algorithm uses a multi-level coarsening of the graph to compute clusters. The method allows as well to set size constraints. We used this algorithm as a baseline for comparison, but due to its limited scalability compared to affinity, we are able to use it only on the smaller publicly-available datasets.

4.3.3 *Step three: post-processing.*

Finally, once we obtain the graph-based clusters (with either methods), we perform a post-processing step which we will show can improve the performance significantly.

Initial cluster-centroid computation. Having generated a clustering, for each such cluster we obtain a centroid by averaging the user's profile vector in the cluster which is associated to the cluster.

Llyod's clustering improvement rounds. Once we compute the cluster centroids we optionally apply some rounds of the Lloyd's iterative method. This is an iterative algorithm which repeatedly assigns points to their nearest centroid, and then computes the mean of each cluster as the centroid. To implement this efficiently we use a SimHash-based locally sensitive hashing scheme to solve the nearest neighbor search problem for finding the nearest center.

Traditionally the Lloyd's method is used to optimize k-means cost by initializing it by random points as centroids. However, we note that the initialization with affinity clustering or METIS is essential for our applications for two reasons: 1) without this initialization, the algorithm will greatly violates the size constraints during the Lloyd local optimization, and 2) the number of rounds to converge to a high-quality solution increases significantly. Nevertheless, we observe that a few iterations of Lloyd algorithm, initialized with affinity or METIS centroids, can increase the quality of the solutions without violating the imposed size constraints of the vast majority of clusters as observed in the empirical study.

Final user-to-cluster assignment. The final step of the post-processing method is to associate each user with a cluster. This is done by simply assigning each user to the cohort corresponding to the nearest centroid. Notice that, in principle, the final cluster assignment might be quite differ from the clusters obtained by the graph-based clustering.

Unlike SimHash or SortingLSH, affinity clustering and METIS use the information of users to actively search for similar users. Therefore, we should expect to get a much better privacy-utility trade-off from these algorithms. Nevertheless, the generation of the user-to-user graph requires access to the full browsing history of a user by a centralized server. This is a trade-off that needs to be weighed by any browser wishing to implement this algorithm.

5 EVALUATION ON PUBLIC DATASETS

To evaluate the quality of the clustering algorithms we measure the ability to group similar users together. We recall that the cosine similarity between two vectors x_1 and x_2 is given by

$$\text{CosSim}(x_1, x_2) = \frac{x_1^\top x_2}{\|x_1\|\|x_2\|} \in [-1, 1].$$

Given a cohort C we define the cohort similarity as

$$\text{Sim}(C) = \frac{1}{|C|} \sum_{x \in C} \text{CosSim}(x, \mu(C)).$$

That is, the cosine similarity of a cohort is the average cosine similarity between all elements in the cohort and the cohort's centroid. Finally, the quality metric for an algorithm is the average cohort similarity of all cohorts generated by the algorithm. Notice that an algorithm that assigns each user to its own cohort will have the perfect quality metric of 1. To ensure we do not favor algorithms that generate small cohorts, we also define a privacy metric.

Let n denote the number of users in a dataset and $U(k)$ denote the number of users in cohorts of size at least k. To assess the privacy properties of each algorithm we look at the following anonymity metric

$$\text{anon-quantile}(\alpha) = \max_{k:U(k)>\alpha n} k$$

That is, the largest k such that α fraction of users belong to a cohort that is k-anonymous. For the results reported in this section we set $\alpha = .98$.

5.1 Movielens 25M

The MovieLens 25M [26] dataset consists of 25 million movie ratings keyed by user id and scored from 0 stars to 5 stars. Each movie is associated with one or more categories chosen from a dictionary of 20 movie genres. These genres include categories like comedy and thriller as well as a "no genres listed" category

User embedding generation. We proceed to describe the process by which we obtain user embeddings in $\mathbb{R}^d$ for the users in the data. The process consists of two steps:

(1) Feature extraction: Given the set of genres G, We encode each (movie, user) pair as a feature vector $v \in \mathbb{R}^{|G|}$ where $v_i > 0$ if and only if the movie is associated with genre i. When $v_i > 0$ its value is given by the ranking of the movie provided by the user.
(2) Feature vector aggregation: To aggregate all feature vectors associated with a user we simply take the average of the vectors.
(3) Centering. Finally, we center all feature vectors to ensure the dataset has mean zero.

The choice of centering the data was made to ensure that a baseline that randomly groups users together on cohorts of the same size has an average cohort similarity of zero.

5.2 Million song dataset

The million song dataset (MSD) is a collection of 1 million songs tagged by categories and user ids. The dataset consists of the listening history of 650 thousand users. Each (user, song) pair is tagged by the number of times it was listened to as well as the categories the song belongs to. These categories have weights representing how well the category describes each song. Examples of these tags are "Pop" or "60's". The dataset also includes some subjective tags such as "good" or "awesome". The original dataset consists of more than 200 thousand categories. However, most of these categories appear only a few times in the whole dataset. For this reason we restricted the set of categories to those who appear in at least 1% of the songs. The user embedding process is very similar to the embedding process described for the MovieLens dataset. The only difference is that the entry v_i of the song feature vector corresponds to the product of the number of times the user listened to that song and the significance of the category to the song.

The results of running the proposed clustering algorithms on both datasets can be found in Figure 6. The first thing to notice is that, not surprisingly, the use of a centralized algorithm improves the quality of the cohorts, specially at high levels of anonymity. In

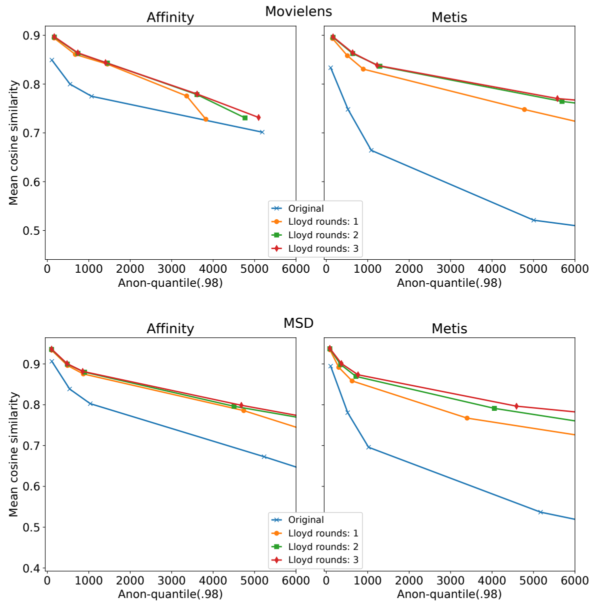


Figure 5: Results on the Movielens and MSD dataset with varying numbers of Lloyd rounds applied to Affinity and Metis clustering.

fact, even when SimHash generates very small cohorts, it seems to not be able to match the performance of affinity clustering at an anonymity level of 1000. This shows that there is certainly much to be gained by using a centralized graph-based algorithm. Nevertheless, it is interesting to see that the biggest gain in quality comes from comparing the fully decentralized SimHash algorithm to a random baseline which would have a cosine similarity of zero. This result is rather surprising and demonstrates that clustering without access to raw user information is still a viable solution to generating cohorts.

To further understand the effect of using multiple rounds of training in our centralized algorithms we show in Figure 5 the quality-privacy tradeoffs of these algorithms after different number of rounds of Lloyd's algorithm there we notice that even though the size of clusters diminishes (the plot shifts a slightly to the left), this minor size constraint violation is overshadowed by the much bigger gain in utility. Although, this gain seems to plateau after 2 iterations of the Lloyd's algorithm.

6 EVALUATION ON ADS DATASET

We now show results of using cohorts for the task of interest based advertising on a proprietary ads dataset. The dataset corresponds to 7 days of ad impression history recorded by an ad tech company. Each record in the dataset contains up to 5 interest categories associated with the publisher's website. Each category is weighed by the relevance of the category to the website. Overall we consider approximately 2000 categories ranging from '/Arts/Literature' to '/Cars/SportCars/Convertibles'. To generate user vectors we simply take the average of all the weighted categories a user visited over

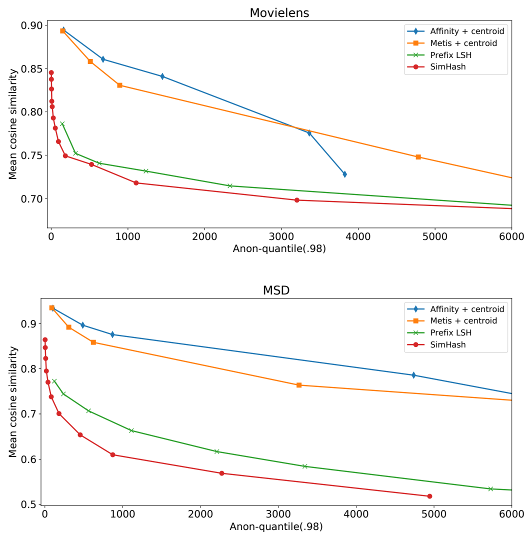


Figure 6: Results on the Million Song and Movielens datasets.

a period of 7 days. To evaluate the quality of the cohorts generated by this algorithm we generate cohort profiles by averaging all categories visited by users in a particular cohort. Finally, we evaluate the quality of these profiles by measuring their ability to predict future conversions. More precisely, let $\mathcal{I}$ denotes the set of interest categories. For a cohort C we denote by $P_C \in \mathbb{R}^{|\mathcal{I}|}$ the cohort profile. For each user U we denote by $P_U \in \{0,1\}^{|\mathcal{I}|}$ a user conversion profile where $P_U[c] = 1$ if user U had a conversion on a website tagged with category c. The user conversion profile is built using 7 days of data as well. The timeline for generating cohorts, cohort profiles and user conversion profiles is depicted in Figure 7. The metrics for evaluating our algorithms will be precision and recall at 10. That is, for a cohort profile P_C, let $\text{Top}(P_C)$ be the top 10 categories (by weight) on profile P_C, for every category c we define an algorithm's precision and recall at 10 as:

$$\text{Prec}[c] = \frac{\sum_C \sum_{U \in C} \mathbb{1}\{P_U[c] = 1 \wedge c \in \text{Top}(P_C)\}}{\sum_C |C| \mathbb{1}\{c \in \text{Top}(P_C) = 1\}}$$
$$\text{Rec}[c] = \frac{\sum_C \sum_{U \in C} \mathbb{1}\{P_U[c] = 1 \wedge c \in \text{Top}(P_C)\}}{\sum_U \mathbb{1}\{P_U[c] = 1\}}.$$

Let $\mathcal{I}' = \{c \in \mathcal{I} | c \in \text{Top}(P_C) \text{ for some C}\}$ denote the set of categories that are in the top 10 categories for at least one cohort profile. We define the overall precision and recall of an algorithm as

$$\text{Prec} = \frac{1}{|\mathcal{I}'|} \sum_{c \in \mathcal{I}'} \text{Prec[c]} \qquad \text{Prec} = \frac{1}{|\mathcal{I}|} \sum_{c \in \mathcal{I}} \text{Rec[c]}.$$

In Figure 8 we show the predictive power of the algorithms proposed here while enforcing 1000-anonymity. Anonymity is enforced by replacing all profiles in cohorts with fewer than 1000 users with the most popular 10 browsing categories. We compare our algorithms against two simple baselines. One that randomly clusters users in groups of 1000. The second corresponds to a case where each user forms a singleton cluster. That is, the precision and accuracy at predicting conversions is based on the current user profiles. Notice that in practice this signal is usually combined with other information such as features of the current website a user is visiting. Thus, the baseline does not exactly match the proprietary approaches of various interest-based advertising platforms; nonetheless we believe it still provides a very strong point of comparison. The values reported are relative to the latter baseline.

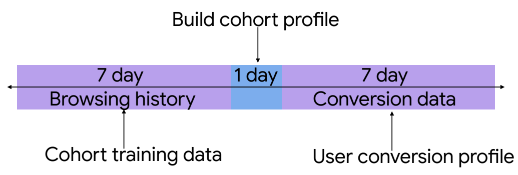


Figure 7: Timeline used to build cohorts, cohort profiles and user conversion profiles. Cohorts are built using 7 days of data. After cohorts are built, the next day of data is used to generate the cohort profiles. Finally the next 7 days are used to generate the conversion profiles.

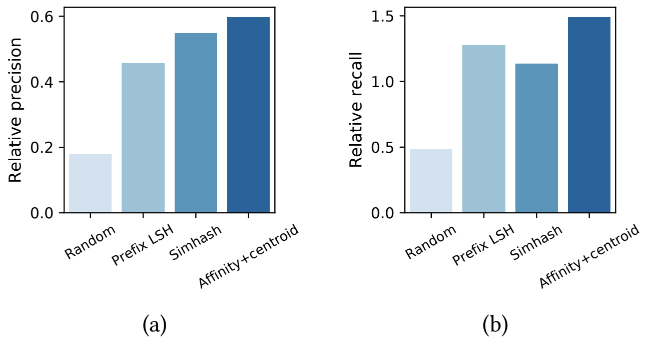


Figure 8: (a) Relative precision on the ads dataset. (b) Relative recall.

The first takeaway from these experiments is that by grouping users with similar interests together we can achieve a better recall than the singleton cluster baseline. This suggests that the FLoC API could be used as a way to expand the reach of campaigns who are trying to target a specific audience. This gain on recall comes with a corresponding decrease in precision. Still, even the fully decentralized SimHash algorithm achieves 55% of the current cookie based profiles or a 4x improvement over a random baseline. These results indicate that interests are preserved through clustering.

7 LIMITATIONS OF THE STUDY

In this paper, we show that there are clustering-based solutions that can achieve certain privacy and utility benchmarks set forth in the paper (namely providing k-anonymity guarantees, while ensuring meaningful signal for interest-based advertising). We stress, however, that our study is only preliminary and that a complete solution for implementing a real-world system within a browser platform that can operate alongside other mechanisms, will need to take into account numerous other considerations that go beyond the scope of this paper. Among the open questions left by this paper, we leave as future work the study of whether differential privacy guarantees can be combined with those of k-anonymity for further enhancing privacy protections. We also leave as future work examining other locally sensitive hashing [14, 43] and clustering [16, 17, 22] algorithms for this application.

8 CONCLUSION

In this paper, we proposed a series of cohort id assignment algorithms that use multiple clustering techniques (distributed and centralized) and extensively evaluated them on different datasets (freely available and proprietary). For the task of predicting conversions, we demonstrated our clustering methods produce an informative signal, and we can achieve significant improvements in recall and precision over randomly assigning users to cohorts even at high anonymity levels. Moreover, the recall of our algorithms can in some cases surpass the recall of naïvely using user profiles for predicting interests. This suggests that the FLoC API could be used to expand the reach of advertisers to similar audiences.

REFERENCES

[1] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. 2005. Approximation algorithms for k-anonymity. *Journal of Privacy Technology (JOPT)* (2005).

[2] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. 2017. Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*.

[3] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. 2009. Streaming k-means approximation.. In *NIPS*, Vol. 4. 2.

[4] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 475–486.

[5] Amitai Armon. 2011. On min–max r-gatherings. *Theoretical Computer Science* 412, 7 (2011), 573–582.

[6] Olivier Bachem, Mario Lucic, and Andreas Krause. 2017. Distributed and provably good seedings for k-means in constant rounds. In *International Conference on Machine Learning*. PMLR, 292–300.

[7] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine learning* 56, 1 (2004), 89–113.

[8] MohammadHossein Bateni, Aditya Bhaskara, Silvio Lattanzi, and Vahab S. Mirrokni. 2014. Distributed Balanced Clustering via Mapping Coresets. In *NeurIPS*. 2591–2599.

[9] Mohammad Hossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, Mohammad Taghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. 2017. Affinity clustering: Hierarchical clustering at scale. In *NeurIPS*. 6867–6877.

[10] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. 2005. LSH forest: self-tuning indexes for similarity search. In *WWW*. 651–660.

[11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

[12] Ji-Won Byun, Ashish Kamra, Elisa Bertino, and Ninghui Li. 2007. Efficient k-anonymization using clustering techniques. In *International Conference on Database Systems for Advanced Applications*. Springer, 188–200.

[13] Sanjoy Dasgupta. 2016. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 118–127.

[14] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *SOCG*, Jack Snoeyink and Jean-Daniel Boissonnat (Eds.). ACM, 253–262.

[15] Patrik D'haeseleer. 2005. How does gene expression clustering work? *Nature biotechnology* 23, 12 (2005), 1499–1501.

[16] Laxman Dhulipala, David Eisenstat, Jakub Łącki, Vahab Mirrokni, and Jessica Shi. 2021. Hierarchical Agglomerative Graph Clustering in Nearly Linear Time. In *ICML*.

[17] Alessandro Epasto, Mohammad Mahdian, Vahab Mirrokni, and Peilin Zhong. 2021. Massively Parallel and Dynamic Algorithms for Minimum Size Clustering. *arXiv preprint* (2021).

[18] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *KDD*, Vol. 96. 226–231.

[19] José Estrada-Jiménez, Javier Parra-Arnau, Ana Rodríguez-Hoyos, and Jordi Forné. 2017. Online advertising: Analysis of privacy threats and protection approaches. *Computer Communications* 100 (2017), 32–51.

[20] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[21] Teofilo F Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *TCS* 38 (1985), 293–306.

[22] John C Gower and Gavin JS Ross. 1969. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 18, 1 (1969), 54–64.

[23] Saikat Guha, Bin Cheng, and Paul Francis. 2011. Privad: Practical Privacy in Online Advertising. In *Proceedings of the 8th USENIX*, David G. Andersen and Sylvia Ratnasamy (Eds.). USENIX Association.

[24] Jonathan Halcrow, Alexandru Mosoi, Sam Ruth, and Bryan Perozzi. 2020. Grale: Designing networks for graph learning. In *KDD*. 2523–2532.

[25] Sariel Har-Peled and Soham Mazumdar. 2004. On coresets for k-means and k-median clustering. In *STOC*. 291–300.

[26] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[27] Daniel C. Howe and Helen Nissenbaum. 2017. Engineering Privacy and Protest: A Case Study of AdNauseam. In *Proceedings of IWPE@SP*, José M. del Álamo, Seda F. Gürses, and Anupam Datta (Eds.), Vol. 1873. CEUR-WS.org, 57–64.

[28] Anil K Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters* 31, 8 (2010), 651–666.

[29] Ari Juels. 2001. Targeted advertising... and privacy too. In *Cryptographers' Track at the RSA Conference*. Springer, 408–424.

[30] Josh Karlin. [n.d.]. Federated Learning of Cohorts (FLoC). https://github.com/WICG/floc

[31] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* 20 (1998), 359–392.

[32] Przemysław Kazienko and Michał Adamski. 2007. AdROSA—Adaptive personalization of web advertising. *Information Sciences* 177, 11 (2007), 2269–2295.

[33] Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *International Colloquium on Automata, Languages, and Programming*. Springer, 597–608.

[34] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. 2019. Fair k-Center Clustering for Data Summarization. In *ICML*.

[35] Jure Leskovec, Kevin J Lang, and Michael Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*. 631–640.

[36] Shi Li and Ola Svensson. 2016. Approximating k-Median via Pseudo-Approximation. *SIAM J. Comput.* (2016).

[37] Gustavo Malkomes, Matt J Kusner, Wenlin Chen, Kilian Q Weinberger, and Benjamin Moseley. 2015. Fast distributed k-center clustering with outliers on massive data. In *Advances in Neural Information Processing Systems*. 1063–1071.

[38] Bashir Muhammad A and Christo Wilson. 2018. Diffusion of User Tracking Data in the Online Advertising Ecosystem. *Proc. Priv. Enhancing Technol.* (2018).

[39] Fionn Murtagh and Pedro Contreras. 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2, 1 (2012), 86–97.

[40] Liudmila Prokhorenkova, Alexey Tikhonov, and Nelly Litvak. 2019. Learning clusters through information diffusion. In *WWW*. 3151–3157.

[41] Pierangela Samarati and Latanya Sweeney. 1998. Generalizing Data to Provide Anonymity when Disclosing Information (Abstract). In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART*, Alberto O. Mendelzon and Jan Paredaens (Eds.). ACM Press, 188.

[42] Jana Schmidt, Andreas Hapfelmeier, Marianne Mueller, Robert Perneczky, Alexander Kurz, Alexander Drzezga, and Stefan Kramer. 2010. Interpreting PET scans by structured patient data: a data mining case study in dementia research. *KAIS* 24, 1 (2010), 149–170.

[43] Anshumali Shrivastava and Ping Li. 2014. In Defense of Minhash over Simhash. In *AISTATS*, Vol. 33. JMLR.org, 886–894.

[44] Statista. [n.d.]. Internet Usage Worldwide. https://www.statista.com/statistics/617136/digital-population-worldwide/

[45] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. 2010. Adnostic: Privacy preserving targeted advertising. In *Proceedings Network and Distributed System Symposium*. The Internet Society.

[46] Dongkuan Xu and Yingjie Tian. 2015. A comprehensive survey of clustering algorithms. *Annals of Data Science* 2, 2 (2015), 165–193.

[47] Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*. 587–596.

[48] Kang Zhao, Hongtao Lu, and Jincheng Mei. 2014. Locality Preserving Hashing. In *AAAI*.