Detecting Users from Website Sessions: A Simulation Study

Corné de Ruijt

Faculty of Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
Email: c.a.m.de.ruijt@vu.nl

Sandjai Bhulai

Faculty of Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
Email: s.bhulai@vu.nl

***Abstract*—In real click data sets, the user initiating a web session may be censored, as unique users are commonly determined by cookies. One way to study the effect of this censoring on various website metrics, and to study the effectiveness of algorithms trying to undo this censoring, is by simulation. We therefore propose a click simulation model, which is capable of simulating user censoring due to cookie churn or the usage of multiple devices, but for which we still keep the uncensored ground truth. To recover unique users from session data, we compare several (H)DBSCAN*-type (Hierarchical Density-based Spatial Clustering of Applications with Noise) algorithms, where we assume that all sessions in a cluster likely originate from the same user. From this comparison, we find that even though the best (H)DBSCAN*-type algorithm does significantly outperform other benchmark clustering methods, it performs considerably worse than when using the observed cookie clusters. I.e., websites for which the assumptions of our simulation model hold, our results suggest that uncovering users from their session data using clustering algorithms may lead to considerably larger errors in terms of user related websites metrics, compared to using cookies to uncover users.**

Keywords–Click models; Session clustering; HDBSCAN*

I. INTRODUCTION

The current Internet environment heavily relies on cookies for the enhancement of our Internet browsing experience. These cookies play a crucial role in session management, the personalization of websites and ads, and user tracking. However, the usage of multiple devices, multiple browsers, and the focus on cookie management, have made the problem of identifying single users over multiple sessions more complex. One study reports that as much as 20% of all Internet users delete their cookies at least once a week, whereas this percentage increases to approximately 30% when considering cookie churn on a monthly basis [1].

Not being able to track Internet users may lead to suboptimal behavior of search engines and online ads, as these have less information about the previous search and click behavior to infer the user's preference for certain items. As cookie churn and the usage of multiple devices cause the user to be censored, we relate to this by the term user censoring.

The problem of unrevealing which session(s) originate(s) from which user has been considered in previous literature, which to our knowledge all have been using real-world data sets to test their algorithms on. Although these real-world data sets have the advantage of capturing much of the complexity of users' click decisions, they also have two clear disadvantages. First, as users are only identified by cookies, we can conclude that two sessions having the same cookie originate from the same user. However, the opposite is not always true: two sessions having different cookies are not per definition different users. The user's cookie might have churned, or the user might have initiated a session from a different device, creating another cookie. Hence, the ground truth is only partially observed in these data sets. Second, using real-world data sets limits the possibility of studying the sensitivity of an algorithm on the underlying click data set: most real-world data sets come from large search engines, which may not always be representative for all websites.

Therefore, we propose a click simulation model, and use realizations of this model to study the effectiveness of several (H)DBSCAN*-type clustering algorithms on uncovering users from their sessions. To measure the effectiveness of these algorithms, we not only consider the error in terms of typical supervised clustering error measures such the adjusted Rand index, but also in terms of the error in estimating overall web statistics, such as the number of unique users, distribution of the number of sessions per user, and the user conversion distribution.

To avoid making the simulation model overly complex, we decided to model user interactions with a search engine. This has two main advantages: first, there exists quite extensive literature on what type of parametric models are accurate for modeling user behavior on search engines [2]. Second, apart from dedicated search engines, a search tool is also a common feature on websites serving other purposes [3]. Another measure against complexity is that we assume all users push homogeneous queries, that is, although we allow users to have different preferences for items returned by the search engine, we do assume all users push the same query.

This paper has the following structure. Section II discusses relevant literature related to session clustering. Section III discusses the simulation model, adaptions of (H)DBSCAN*, and the experimental setup. Section IV discusses the results, whereas Section V discusses the implications of these results and ideas for further research.

II. RELATED WORK

Simulating click behavior is not a new concept: Chucklin et al. [2][pp. 75-77] suggest using pre-fitted click models for this purpose, where the model is pre-fitted to public click data sets. One risk of using pre-fitted models is an availability bias: can the characteristics of public click data sets, commonly provided by large search engines, easily be generalized over all search

 ISBN: 978-1-61208-816-7

engines? Also, these data sets do not always provide the type of information one is interested in, such as the device used to initiate a session.

Fleder and Hosanagar [4] provide a generative approach for modeling user preferences, which we will discuss in more depth in Section III-A. This model can be used as an alternative to model user preferences. Pre-fitted and generative models do have a trade-off in terms of accuracy vs interpretability. I.e., pre-fitted models may have an accurate estimate of user item preferences, but provide little understanding in why this preference over different items has a certain shape, whereas generative models do explain why a user has a certain item preference, but these models might be less accurate.

Several authors have studied how cookie censoring occurs. For example, [1][5][6] consider cookie censoring due to cookie churn, whereas [7] considers cookie censoring due to using multiple devices. Results from these studies can be used to model cookie churn dynamics in a simulation model.

Identifying unique users from sessions can be seen as a specific case of the entity/identity resolution problem [8]. Though what makes this problem special is the nature of the data set. This typically consists of a large number of sessions, for which clicks and web page meta-data (such as the URL) are logged. Because of these characteristics, entity resolution algorithms that do not account for these characteristics are likely to fail in their objective. Karakaya et al. [9] give a survey of the literature on cross-device matching, i.e., where it is assumed that user censoring occurs as users use multiple devices. However, many of the approaches listed can also be applied to more general settings.

The problem of uncovering users from their sessions obtained considerable scientific attention following the 2015 ICDM and 2016 CIKM machine learning challenges [9]–[11]. Interestingly, although at a first glance much of the literature seems to relate to the same problem, the context of the data, and how the problem is interpreted seems to vary greatly. The 2015 ICDM and 2016 CIKM challenges consider the problem from the perspective of an online advertiser, where data is gathered from multiple websites. Others consider the problem from a single website perspective [1][8][12]. Although the underlying problem may be the same from both perspectives, the solution may not. E.g., since in the advertisement case the data set contains a variety of URLs from different websites, these solutions rely more on natural language processing techniques than in the single website case.

There also seems to be ambiguity in whether the solution should allow for overlapping session clusters. Most commonly (like in the 2015 ICDM and 2016 CIKM challenges), the problem is modeled as a binary classification problem, predicting whether pairs of sessions originate from the same user [9]. As a result, this interpretation of the problem does allow for overlapping session clusters. Other approaches restrict themselves to non-overlapping clusters, but do however have other disadvantages, like computational feasibility for large data sets [13], or additional assumptions about user behavior [1].

III. METHODS

A. Simulating click data with cookie-churn

We will describe the simulation model in three parts. The first part models how users navigate through a single Search Engine Result Page (SERP), for which we use a click model. The second part models how user preferences over different items are determined, while the third part models how a session's underlying user is censored due to cookie churn or the usage of multiple devices.

To describe the simulation model, the following notation will be used. Let $i \in \{1, \ldots, n\}$ be a query-session, which produces a SERP of unique items $\mathcal{L}_i \subseteq \mathcal{V}$, with $\mathcal{V} = \{1, \ldots, V\}$ the set of all items, indexed by v. A (query-)session consists of a sequence of interactions with a single SERP, and these interactions are completely defined by the click model. Let $u_i \in \mathcal{U}$ denote the user initiating query-session i, with $\mathcal{U} = \{1, \ldots, U\}$ the set of all users. The user index u is used instead of u_i in case the precise query-session i is irrelevant.

1) Simulating SERP interactions: To simulate clicks on a search engine, we employ the Simplified Dynamic Bayesian Network model (SDBN) [14]. The main reason for choosing SDBN is that, though the model is simple, it seems to perform reasonably well in comparison to other parametric click models when predicting clicks [2]. The two main variables in this model are, for all $u \in \mathcal{U}$, $v \in \mathcal{V}$, the probability of attraction $\phi^{(A)}_{u,v}$ (probability of user u clicking item v, given that v is evaluated), and the probability of satisfaction $\phi^{(S)}_{u,v}$ (probability of user u evaluating an item at position $l+1$ in the SERP, given that item v at position l was just clicked). SDBN assumes that the first item in a SERP is always evaluated.

2) User item preferences: To come up with reasonable values for $\phi^{(A)}_{u,v}$ and $\phi^{(S)}_{u,v}$, we use the same approach as in [4], which we will refer to as Fleder-Hosanagar's model. That is, each user $u \in \mathcal{U}$ and each item $v \in \mathcal{V}$ is represented by the vectors $\boldsymbol{\eta}_u = \left(\eta^{(u)}_1, \eta^{(u)}_2\right)$, and $\boldsymbol{\psi}_v = \left(\psi^{(v)}_1, \psi^{(v)}_2\right)$ respectively, where both are drawn i.i.d. from a standard bivariate normal distribution. The probabilities $\phi^{(A)}_{u,v}$, and $\phi^{(S)}_{u,v}$ are then determined by the multinomial logit:

$$\phi^{(X)}_{u,v} = \frac{e^{\omega_{u,v}+\nu^{(X)}}}{\sum_{v' \in \mathcal{V}\setminus\{v\}} e^{\omega_{u,v'}} + e^{\omega_{u,v}+\nu^{(X)}}}, \qquad (1)$$

with $\omega_{u,v} = -q \log \delta(\boldsymbol{\eta}_u, \boldsymbol{\psi}_v)$, and $X \in \{A, S\}$. Here δ is some distance function, in our case Euclidean distance. $q \in \mathbb{R}^+$ is some constant value that models the user preference towards nearby products, and $\nu^{(A)}$, $\nu^{(S)}$ are salience parameters for attraction and satisfaction, respectively.

3) User censoring: User censoring is incorporated in the simulation model in two ways: by letting cookies churn after some random time T, and by switching from device d to some other device d'. First, we consider the cookie lifetime $T^{\text{cookie}}_{u,o,d}$ for the o-th cookie of user u on device d, and the user lifetime T^{user}_u. Whenever the cookie lifetime of cookie o ends, but the current user lifetime is strictly smaller than T^{user}_u, a new cookie o' is created, for which the lifetime is drawn from the cookie lifetime distribution F^{cookie}. For a period of $T^{\text{cookie}}_{u,o',d}$, all click behavior of user u on device d will now be registered under cookie o'.

Second, after each query-session, a user may switch from device d to d', which happens according to transition matrix P. Whenever a user switches devices, we consider whether the user has used this device before. If not, a new cookie o' is created, and we draw a new cookie lifetime from F^{cookie}.

However, the cookie lifetime $T^{\text{cookie}}_{u,o,d}$ does not end prematurely when the user switches from device d to d'. If later on the user switches back to device d while the cookie lifetime $T^{\text{cookie}}_{u,o,d}$ has not ended, the behavior of user u is again tracked via cookie o until another device switch occurs or cookie o churns.

Putting this censoring into practice requires us to provide five distributions: 1) a distribution F^{abs} for the time between query-sessions, which following [6] we will refer to as the *absence time*, 2) a distribution for the cookie lifetime (F^{cookie}), 3) a distribution for the user lifetime (F^{user}), 4) the device transition matrix P, and 5) the initial device probability $\boldsymbol{\pi}$. All distributions were adopted from previous literature, which is summarized in Table I.

TABLE I. DISTRIBUTIONS REGULATING COOKIE CHURN.

Variable	Description	Distribution
$T^{\text{cookie}}_{u,o,d}$	Cookie lifetime o-th cookie of user u on device d	Hyper-exponential [1]
$T^{\text{abs}}_{u,i}$	Time between sessions i and $i+1$ of user u	Pareto-I, fitted using data from [6]
T^{user}_{u}	Lifetime of user u	Sum of N_u hyper-exponentials [1], $N_u \sim geom(\rho)$
P, $\boldsymbol{\pi}$	Device transition matrix and initial transition probabilities	From [7], removing the game console device from the state space

4) Summary of the simulation procedure: By combining the three simulation modules, we obtain the full simulation procedure. In short, we first simulate attraction and satisfaction parameters $\phi^{(A)}_{u,v}$, $\phi^{(S)}_{u,v}$ for all (u, v) pairs using Fleder-Hosanagar's model. Second, we simulate clicks for a set of $\mathcal{U}_{\text{warm-up}}$ users (where $\mathcal{U}_{\text{warm-up}} \cap \mathcal{U} = \emptyset$) using SDBN, where the item order for each query-session is determined uniformly at random. Third, we again run SDBN, now incorporating user censoring, over the set $\mathcal{U}$. The item order in each query-session is now draw i.i.d. from a multinomial distribution (without replacement), where the probabilities are proportional to the overall item popularity found during the warm-up phase. The simulation's source code is available via Github [15].

Although so far we assumed all users arrive at $t = 0$, we shift all times after the simulation to obtain click behavior spread out over time. Here, we assume a Poisson arrival process with rate γ. I.e., the first query-session of user u starts some exponentially distributed time after the initial query-session of user $u-1$. Note that these inter-first session times only depend on the time of the first session of the previous user, not on any other subsequent behavior of that user.

B. Session clustering

1) Introducing maximum cluster sizes to HDBSCAN and DBSCAN*:* Due to space limitations, we will refer to [16][17] and [18] for details on how the DBSCAN* and HDBSCAN* algorithms work. What we will use, is that both algorithms initially represent the data in a dendogram. The dendogram is obtained by computing a Minimum Spanning Tree (MST) over the complete weighted graph of pairwise distances between data points (in our case sessions), which causes a considerable speed improvement compared to many other hierarchical clustering methods.

As the data is represented in a dendogram, all data points are at the leaves of this binary tree. This tree has the property that if the shortest path between two leave nodes has to use an edge closer to the root node, then these two data points are further apart. This implies that, if we perform a horizontal cut on the tree, this cut relates to some maximum distance ϵ, and all branches below this cut share the property that all data points connected in this branch are at most ϵ apart. Leaf nodes above the ϵ-cut are labeled as noise, and obtain their own cluster.

To impose a maximum cluster size, we employ three strategies, which we name MS-DBSCAN*, MS-HDBSCAN*$^-$ and MS-HDBSCAN*$^+$ (the abbreviation 'MS' stands for 'Maximum Size'). In MS-DBSCAN*, we first make the ϵ-cut to obtain branches $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$. Next, if some branch τ_j contains more than β data points, we split the branch again at the root node: considering the branches at the left and right child of the root of τ_j as potential clusters. This splitting is continued until all branches have fewer than β data points, after which we assign all data points in one branch to the same cluster.

Both MS-HDBSCAN*$^-$ and MS-HDBSCAN*$^+$ first perform HDBSCAN* using the relative excess of mass to split the dendogram into different branches. Next, we apply the same strategy as in MS-DBSCAN*, where we continue splitting the branches until all branches have fewer than β data points. As a last step, we rerun HDBSCAN* separately on the individual branches, which may again split a branch into smaller branches if that improves the relative excess of mass over all resulting clusters. The only difference between MS-HDBSCAN*$^-$ and MS-HDBSCAN*$^+$ occurs when some found branch (in other words, cluster) does not make any splits for which the left and right child have at least M data points, for some given $M \in \mathbb{N}$. In MS-HDBSCAN*$^+$, we consider all points in this branch to be contained in the same cluster, whereas MS-HDBSCAN*$^-$ assumes all data points in this branch are noise points.

2) Session cluster re-evaluation: As one might have noticed, insofar we have not used any information from the cookies. I.e., knowing which sessions have the same cookie could provide valuable information about the underlying user. In particular, we wish to train a model that can function as an alternative to the standard distance measure δ in (H)DBSCAN*, such as Euclidean or Manhattan distance, which we then again can plug into the adapted (H)DBSCAN* algorithms.

Obtaining session clusters with re-evaluation is done as follows. Assume we have a trained classifier $\hat{f}(X_i, X_{i'})$, which returns the probability of sessions X_i and $X_{i'}$ originating from the same user. First, like in [19], we find for each point X_i the K nearest neighbors, which gives us a set $\mathcal{X}$ of all nearest neighbor session pairs. Second, we compute $-\log(\hat{f}(X_i, X_{i'}))$ for all $(X_i, X_{i'}) \in \mathcal{X}$, and fill this into a (sparse) $n \times n$ distance matrix W. For all pairs $(X_i, X_{i'}) \notin \mathcal{X}$, we assume the distance is some large value $\delta_{\max}$, which allows us to store W efficiently, and greatly speeds up computations compared to evaluating all pairwise same user probabilities. Distance matrix W can subsequently be used as distance measure δ in the algorithms discussed in Section III-B to obtain new session clusters.

We use a logistic regression model for $\hat{f}$, which we train by undersampling from a set $\mathcal{X}_{\text{clust}} \cup \mathcal{X}_{\text{cookie}}$, where $\mathcal{X}_{\text{cookie}}$ contains all pairwise sessions sharing the same cookie cluster, and $\mathcal{X}_{\text{clust}}$ is a set of all pairwise sessions sharing the same computed

 ISBN: 978-1-61208-816-7

cluster. These computed clusters are obtained by running a (H)DBSCAN*-type algorithm using Euclidean distances.

3) DBSCAN with random clusters:* To benchmark the clustering approaches just discussed, we consider the following benchmark. We first cluster the sessions using the ordinary DBSCAN* algorithm, in which way we obtain initial clusters $\mathcal{B}_1^0, \ldots, \mathcal{B}_m^0$. Next, for each cluster $\mathcal{B}_j^h$ ($h \in \mathbb{N}_0$, with initially $h = 0$), if $|\mathcal{B}_j^h| > \beta$, we iteratively select $\min\{s_{j,h}, |\mathcal{B}_j^h|, \beta\}$ points uniformly at random from $\mathcal{B}_j$ to form a new cluster $\widetilde{\mathcal{B}}$, and update $\mathcal{B}_j^{h+1} \leftarrow \mathcal{B}_j^h \setminus \widetilde{\mathcal{B}}$. Here, $s_{j,h}$, $j = 1, \ldots, m$; $h = 0, 1, \ldots$; are drawn from a geometric distribution with $p = 0.5$. This process continues until for all $j \in \{1, \ldots, m\}$: $|\mathcal{B}_j^h| \leq \beta$ for some h, at which the remaining points in $\mathcal{B}_j^h$ are labeled as one cluster.

Intuitively, we select this benchmark as it captures the higher-level hierarchy clustering of DBSCAN*, but not the low-level clusters (as these clusters are picked at random). Therefore, comparing the previous methods with this random clustering approach allows us to assess whether the smaller size clusters reveal more information than the larger ones.

C. Experimental setup

1) Simulation parameters: Our experimental design consists of two steps. First, we consider a simulation base case on which we evaluate the clustering approaches discussed in Section III-B. In this base case, the users' first query arrival follows a Poisson process with rate $\gamma = 0.2$ (minutes), after which subsequent behavior over time of a particular user is modeled according to F^{abs}, F^{cookie}, F^{user}, F^{device}, P, and $\boldsymbol{\pi}$, of which the parameters were already given in Section III-A3. We use $U = 20,000$ users and $U_{\text{warm-up}} = 2,000$ warm-up users.

Furthermore, we remove the first 250 sessions (not part of the first $U_{\text{warm-up}}$ users, who are only used to estimate the overall item popularity), as these are likely to all be first sessions from newly arriving users, and therefore including them may lead to a bias in the data. Likewise, we remove all observations after $43,200$ minutes (30 days) to avoid the opposite bias: not having any new users. Users could pick from $V = 100$ items, and we choose as maximum list size $L = 10$.

For parameters that could not be adopted from the literature, we tried several parameter values. We find that $q = 1$ (user preference for nearby products), $\rho = 0.5$ (geometric parameter for the number of user lifetime phases N_u), and salience parameters $\nu^{(A)} = \nu^{(S)} = 5$ are reasonable for our base case. In the second step of the experimental design, we make adjustments to the latter parameters, that is, those not adapted from the literature.

2) Features and MS-(H)DBSCAN hyper-parameter settings:* The simulated data set is split into a training and test set according to a 70/30 split over the users. For each session, we use the session's *start time*, *observed session count* (as observed by the cookie), *number of clicks*, and whether the session's *SERP has at least one click* as features. Furthermore, to obtain a vector representation of the items and interactions with the SERP, we first compute a bin-count table. This table contains per item the *total number of clicks, skips (no click)*, and the *log-odds ratio between clicks and skips* over 30 percent of all training sessions, which combined are used as item vector representations. For each SERP, we subsequently concatenate the item vector representations based on their position in the SERP, and multiply this vector with the vector of clicks, vector of skips (=no click), and a hot vector of the last clicked item. The resulting three vectors are, together with the features mentioned earlier, concatenated to obtain the final session vector.

For each method, we experiment with (H)DBSCAN's k-NN parameter, for which we tried $k \in \{1, 3, 5\}$. For DBSCAN*-like algorithms, we try

$$\epsilon \in \left\{ \left(q_{\max} \left(q_{\min} / q_{\max} \right)^{\ell / N} \right) \middle| \ell \in \{1, \ldots, N\} \right\}, \quad (2)$$

with $N = 9$ and $q_{\min}$, $q_{\max}$ the minimum and maximum Euclidean distance between all session pairs of 1,000 sampled sessions. For HDBSCAN*-type algorithms, we set the minimum cluster size to $M = 2$. To train classifier $\hat{f}$, we first run MS-DBSCAN* with the best found values for k and ϵ from earlier validation of MS-DBSCAN* on the training set to, together with the cookie clusters, obtain $\mathcal{X}_{\text{train}}$. Next, we compute the Manhattan and Euclidean distances, and infinity norm between the session vectors of each pair $(i, i') \in \mathcal{X}_{\text{train}}$, which are used as feature vector to train a logistic regression model. We select for each point the $K = 1,000$ nearest neighbors to evaluate the classifier $\hat{f}$ on. All non-evaluated pairs receive distance $\delta_{\max} = -\log\left(10^{-6}\right)$. Next, the MS-(H)DBSCAN* algorithms are evaluated using the new distance matrix W, where we experiment again with $k \in \{1, 3, 5\}$, and

$$\epsilon \in \left\{ q_{\min} + \frac{\ell (q_{\max} - q_{\min})}{N_{\text{re-eval}}} \middle| \ell \in \{1, \ldots, N_{\text{re-eval}}\}, \right\}, \quad (3)$$

using $N_{\text{re-eval}} = 5$.

3) Error metrics: We considered error metrics from two perspectives. First, we consider error measures with respect to the overall website performance. More precisely, given some final clustering $\{\mathcal{B}_1^{\text{final}}, \ldots, \mathcal{B}_m^{\text{final}}\}$, the following error measures are computed. 1) We compute the APE (absolute percentage error) between the real and estimated number of unique users (the latter being equal to m), 2) the Kullback-Leibler divergence (KL-divergence) between the real and estimated user session count distribution (the latter being equal to the cluster size distribution), and 3) the KL-divergence between the real and estimated user conversion distribution. Here, user conversion is defined as the fraction of items clicked per user over all shown (but not necessarily evaluated) items.

The second perspective is on the level of the clusters themselves, where we consider two error measures. To determine the quality of the clusters, we computed the adjusted Rand index (ARI) [20] between computed and real session clusters. Besides ARI, we also measure how well the model distinguishes whether each new session originates from an existing or already observed user, which is measured using the accuracy score. Since ARI measures the overlap between the computed and real session clusters, we consider ARI to be our main error measure.

IV. RESULTS

A. Results on the base simulation case

Table II shows how the different models perform in terms of several error measures on both the train and test set. For each method, the shown results are the best results obtained under

the different hyper-parameters tried for that method under that data set. I.e., in theory the hyper-parameters might be slightly different between training and test, though in practice we found this was rarely the case.

TABLE II. RESULTS ON THE BASE CASE.

Model	Data set	ARI	KL-div. session count	KL-div. conversion	APE unique user	New user accuracy
MS-DBSCAN*	train	0.0012	**0.55**	0.13	15	**0.56**
MS-DBSCAN*$_p$	train	**0.14**	0.74	**0.092**	77	0.5
DBSCAN*-RAND	train	0.0002	1	0.096	**0.011**	0.42
MS-HDBSCAN*$^+$	train	0.0007	0.75	0.15	10	0.52
MS-HDBSCAN*$^-$	train	0.0007	0.75	0.15	10	0.52
MS-HDBSCAN*^+_p	train	0.092	0.9	0.11	**0.011**	0.46
MS-HDBSCAN*^-_p	train	0.1	0.9	0.11	**0.011**	0.46
OBS	*train*	*0.91*	*0.017*	*0.0032*	*15*	*0.95*
MS-DBSCAN*	test	**0.0022**	**0.11**	**0.0026**	60	**0.56**
MS-DBSCAN*$_p$	test	0.0015	1.4	0.13	**6.8**	0.4
DBSCAN*-RAND	test	0.0004	0.32	0.015	40	0.5
MS-HDBSCAN*$^+$	test	0.002	0.16	0.0042	53	0.55
MS-HDBSCAN*$^-$	test	0.002	0.16	0.0042	53	0.55
MS-HDBSCAN*^+_p	test	0.0015	1.4	0.13	7.2	0.4
MS-HDBSCAN*^-_p	test	0.0015	1.4	0.13	7.2	0.4
OBS	*test*	*0.91*	*0.1*	*0.0076*	*51*	*0.95*

The *OBS* model in the table are the scores one would obtain if the observed cookies would be used as clusters. Models using the classifier as distance measure are indicated using subscript p. What immediately becomes apparent is that, compared to these observed cookie clusters, all methods perform considerably worse. Hence, in the scenario we consider: a single query where the true location $\boldsymbol{\eta}_u$ is only revealed by clicked and skipped item locations, our approaches do not come near what one would obtain if one would simply take the observed cookies.

However, the scores do reveal some interesting patterns. First, approaches using a probabilistic distance measure seem to overfit the data: they perform relatively well (compared to the other approaches) on various measures on the training set, but on the test set these results are mitigated: here MS-DBSCAN* seems to work best when considering multiple error measures. Looking at the results from different hyper-parameter settings for MS-DBSCAN*, we observe that selecting $k = 1$ performed best. Furthermore, due to our maximum size constraint, the clusters did not alter for $\ell \geq 4$ (corresponding to $\epsilon \geq 6.33$).

Furthermore, methods without a probabilistic distance measure do outperform the DBSCAN*-RAND method on most measures. I.e., they perform better at picking sessions originating from the same user given high-level clusters, than if we would pick session pairs at random from these high-level clusters. Although it is difficult to draw a firm conclusion, these findings might be an indication that the same user signal we try to infer from the click data is somewhat weak: if our methods would not pick up a signal at all, we would expect them to have the same result as the DBSCAN*-RAND method.

B. Results on multiple simulation scenarios

In order to judge the sensitivity of our findings on the parameter settings of the simulation model, we permute the simulation settings to see if this alters our results. As re-running all models on all simulation settings would be computationally rather expensive, we only re-evaluate the best performing models on the simulation cases. Since in our base case we found that the parameters $k = 1$, and $\epsilon = \left(q_{\max}\left(q_{\min}/q_{\max}\right)^{2/3}\right)$ work reasonably well, these parameters are used for MS-DBSCAN* and DBSCAN*-RAND. The maximum cluster size remains the same as in the base case.

Fig. 1 shows how the models perform over the different simulation settings in terms of ARI, which is our main response variable. The figure suggests that all cluster models do stochastically dominate DBSCAN*-RAND. Furthermore, MS-DBSCAN* seems to outperform the other clustering methods in terms of ARI. As assumptions like homogeneity of variance or normality do not hold in this case, we used a Kruskall-Wallis test, which rejects in this case that all median ARI scores over the different methods are the same for any reasonable significance level (e.g., $\alpha = .01$, $p < 10^{-4}$). Pairwise one-sided Wilcoxon signed rank tests between MS-DBSCAN* and all other methods also indicate that MS-DBSCAN* performs significantly better than the other methods (all p-values are smaller than 10^{-4}).

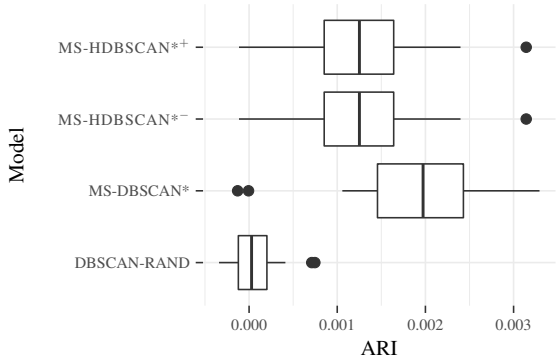

Figure 1. Scores over all simulations.

Considering the variance in the ARI scores, we noticed that strengthening the signal, that is, increasing click probabilities, leads to some improvement in ARI. The most obvious way to do so is by decreasing the number of items (which, as we use bin-counting, ensures each item has sufficient data for bin-counting). However, these improvements remain small. Also interesting is that, when correlating the different error scores over all simulation cases, ARI seems to be weakly correlated with most other error measures, with the sign being in the desired direction (i.e., decrease in KL-divergence for both session count and conversion, but an increase in the new user accuracy). However, improved APE for the number of unique users seems to lead to worse performance in terms of ARI and new user accuracy (Pearson correlations 0.38, and 0.95 resp.).

V. Conclusion and Discussion

In this paper, we presented a homogeneous query click simulation model, and illustrated its usage to the problem of uncovering users from their web sessions. The simulation model is composed of several parametric models, of which previous literature suggests that these models work well in explaining typical patterns observed in click data, while remaining relatively simple. Such patterns include the position bias, cookie censoring, and user preference over multiple products.

 ISBN: 978-1-61208-816-7

Furthermore, we illustrated the simulation model on the problem of (partially observed) session clustering, that is, identifying unique users from their query-sessions. To solve this problem, we tested several mutations of (H)DBSCAN*, where these mutations differ from HDBSCAN* or DBSCAN* as they allow for incorporating a maximum cluster size. From comparing these (H)DBSCAN*-type algorithms, we found that the accuracy of using cookies largely outperform that of not or partially using cookie data. This considerable difference seems to be due to two reasons. 1) The simulated censored cookies turned out to be rather accurate, implying that, assuming the parameters used for cookie censoring adapted from previous literature are accurate, censoring in cookie data does not impose that much of a problem in accurately measuring the metrics studied in this paper. 2) As we only considered a homogeneous query, the user preferences are only revealed from the items users clicked, a signal the various (H)DBSCAN*-type algorithms found difficult to detect. Strengthening this signal, e.g., by increasing the number of clicks, led to a small improvement in ARI.

Other interesting observations include the difference between using Euclidean distance and a probability distance measure in the (H)DBSCAN*-type algorithms, the latter being obtained from training a classifier on detecting whether session pairs originate from the same user. The results suggest that the probabilistic classifier tends to overfit. Also interesting is that, when considering the correlations between the various error metrics considered in this paper, we observed that some error measures show contradictory correlations. In particular, the positive correlation between cluster ARI and average percentage error in the number of unique users (.38), and between the accuracy in estimating whether the next session originates from a new user and the new user average percentage error (.95), indicate that optimizing for one of these error measures may lead to decreased performance in the other.

Although our findings suggest that the practicality of session clustering from single query click data is limited, the usage of the simulation model did allow for studying the sensitivity of the clustering algorithms on different click behavior, something that would not easily have been possible with real click data. It also allowed us to study the effects of user censoring caused by cookie churn or the usage of multiple devices, which showed that if we adopt models of cookie churn behavior found in the literature, this censoring only has a small negative effect on the accuracy of the website metrics discussed in this paper, with an exception for estimating the number of unique users.

Given our findings, a number of questions remain. First, it would be interesting to extend the simulation model to allow for multiple queries. As the solutions to the (multi-query) CIKM 2016 and ICDM 2015 cross-device matching competitions were quite successful, a logical hypothesis would be that incorporating multiple queries into the simulation model would improve the results obtained from (H)DBSCAN*-type algorithms. Second, in this study, we only used a logistic regression model to approximate the probability of two sessions originating from the same user. Given the limited success of this approach so far, it would be interesting to consider other approaches. As the limited results seem to be due to overfitting, including regularization or using bootstrap aggregation approaches could lead to better results. Third, there is still limited knowledge on how cookie censoring occurs. Currently, multiple models exist in the literature, but most models only consider a specific type of censoring, from which one cannot infer how these different types of censoring interact.

REFERENCES

[1] A. Dasgupta, M. Gurevich, L. Zhang, B. Tseng, and A. O. Thomas, "Overcoming browser cookie churn with clustering," in Proceedings of the fifth ACM international conference on Web search and data mining. ACM, 2012, pp. 83–92.

[2] A. Chuklin, I. Markov, and M. d. Rijke, Click models for web search. Morgan & Claypool Publishers, 2015.

[3] C. Luna-Nevarez and M. R. Hyman, "Common practices in destination website design," Journal of destination marketing & management, vol. 1, no. 1-2, 2012, pp. 94–106.

[4] D. Fleder and K. Hosanagar, "Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity," Management science, vol. 55, no. 5, 2009, pp. 697–712.

[5] D. Coey and M. Bailey, "People and cookies: Imperfect treatment assignment in online experiments," in Proceedings of the 25th International Conference on World Wide Web. ACM, 2016, pp. 1103–1111.

[6] G. Dupret and M. Lalmas, "Absence time and user engagement: evaluating ranking functions," in Proceedings of the sixth ACM international conference on Web search and data mining. ACM, 2013, pp. 173–182.

[7] G. D. Montanez, R. W. White, and X. Huang, "Cross-device search," in Proceedings of the 23rd ACM International Conference on Information and Knowledge Management. ACM, 2014, pp. 1669–1678.

[8] D. Jin, M. Heimann, R. Rossi, and D. Koutra, "node2bits: Compact time-and attribute-aware node representations," in ECML/PKDD European Conference on Principles and Practice of Knowledge Discovery in Databases, Proceedings, Part 1, 2019, pp. 483–506.

[9] C. Karakaya, H. Toğuç, R. S. Kuzu, and A. H. Büyüklü, "Survey of cross device matching approaches with a case study on a novel database," in 2018 3rd International Conference on Computer Science and Engineering (UBMK), 2018, pp. 139–144.

[10] ICDM, ICDM 2015: Drawbridge Cross-Device Connections, 2015, https://www.kaggle.com/c/icdm-2015-drawbridge-cross-device-connections, retrieved: August, 2020.

[11] CIKM, CIKM Cup 2016 Track 1: Cross-Device Entity Linking Challenge, 2016, https://competitions.codalab.org/competitions/11171, retrieved: August, 2020.

[12] S. Kim, N. Kini, J. Pujara, E. Koh, and L. Getoor, "Probabilistic visitor stitching on cross-device web logs," in Proceedings of the 26th International Conference on World Wide Web. ACM, 2017, pp. 1581–1589.

[13] F. M. Naini, J. Unnikrishnan, P. Thiran, and M. Vetterli, "Where you are is who you are: User identification by matching statistics," IEEE Transactions on Information Forensics and Security, vol. 11, no. 2, 2016, pp. 358–372.

[14] O. Chapelle and Y. Zhang, "A dynamic bayesian network click model for web search ranking," in Proceedings of the 18th international conference on World wide web. ACM, 2009, pp. 1–10.

[15] C. de Ruijt and S. Bhulai, SDBNSimulator, 2020, https://github.com/cornederuijtnw/SDBNSimulator, retrieved: August, 2020.

[16] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 2013, pp. 160–172.

[17] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 10, no. 1, 2015, pp. 1–51.

[18] L. McInnes and J. Healy, "Accelerated hierarchical density clustering," arXiv preprint arXiv:1705.07321, 2017.

[19] M. C. Phan, Y. Tay, and T.-A. N. Pham, "Cross device matching for online advertising with neural feature ensembles: First place solution at CIKM cup 2016," arXiv preprint arXiv:1610.07119, 2016.

[20] L. Hubert and P. Arabie, "Comparing partitions," Journal of classification, vol. 2, no. 1, 1985, pp. 193–218.

 ISBN: 978-1-61208-816-7