Cookieless Cross-Device Matching System

Haoran Li, Yu Zhu, Yang Zhao

Abstract

As a recent trend of more devices used by consumers to access web, online advertising companies are focusing on an integrated advertising program to reach the same user across multiple digital devices like smartphones, tablets and laptop computers. Therefore, cross matching devices that belong to the same users is crucial for the online advertisers. In this paper we describe a similarity-based approach to cross-matching devices based on user behavioural and physical attributes, without accessing users' private data. We have also proposed three evaluation criteria to check the sanity of results in the context of unsupervised data.

Keyword: cross-device matching, similarity, locality sensitive hashing, unsupervised learning, min-hashing, weakly connected component

1 Introduction

The ability to track users and their online habits is very important for advertising companies. By cross matching devices that belong to the same user, the online advertisers can target their audiences more precisely and thus increase the conversion rate. Traditionally, deterministic device matching is used which involves collecting private data [3], and using that data to connect users across devices through a login system. This approach allows publishers and advertisers to know exactly who the user is across their devices, as long as the user is logged in. However, the companies engaging in this device pairing use private data, and users have legitimate concerns about how this data is being used, stored, and shared or sold. Instead, the similarity-based probabilistic device pairing method, including robust de-anonymization introduced in [5] and [4] is more frequently to make a prediction about users. Although the prediction based on probabilistic pairing cannot be 100% certain, as the algorithms get smarter and identify more data points and trends, the probability that the paired devices belong to the same user increases dramatically. In our paper, we will introduce another similarity-based device grouping method which uses unsupervised web request and beacon data of sessions.

The full pipelines of our approach can be found in Figure 1.

2 Data Description

We used data provided by Yume. The data provided consist of two sources: web request data and web beacon data. Request data mainly records the advertisement specific and device specific data, and beacon data are fetched from the player plugins and records more watching behavioral data. We have 1 week's data which is around 3 Terabytes. We carry out

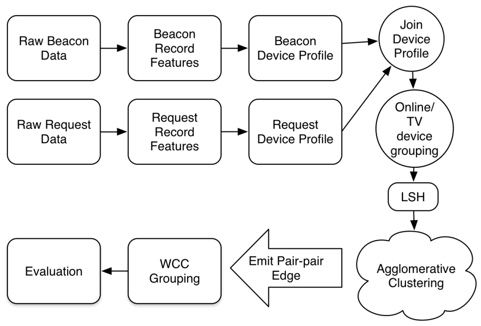

Figure 1: System Pipeline

our pipeline on California data. Since sessions from different states are not likely to be from the same user, it's quite easy to serialize the whole pipeline among all states.

3 Methods and Algorithms

3.1 Feature Selection

Yume beacon and request data consist of 90 features, including domain_id, advertisement_id, view_tracker_percentage, etc. We heuristically selected 50 features that are believed to be related with the identity of a user. Then the correlation analysis was used to further refine our feature list.

3.1.1 Heuristic Feature Selection

The goal of this project is to group all types of devices that belong to the same user, so we need select features that are:

- user-behavior dependent
- uncorrelated with evaluation metrics

According to these criteria, we chose features as shown in 1.

3.1.2 Correlation based Feature Filtering

In addition, we need to filter out features that are highly correlated with the type of device (online, mobile, tablet, smart TV). As including those features will tend to increase the probability of grouping same type device together, which is unlikely for an ordinary user.

We will measure the correlation between each feature with delivery point to filter out the highly correlated ones. In doing so, we use the "cross-type-ratio", which is defined as:

$$\text{cross-type-ratio} = \frac{\begin{array}{c}\text{\# of possible vals mapping to}\\ \text{more than one device type}\end{array}}{\text{Total \# of possible vals}}$$

Feature Group	Feature List
Common	domain_id, placement_id , advertisement_id, census_DMA, publisher_id, content_video_id, service_provider_id, key_value, player_location_id, player_size_id, page_fold_id, ad_visibility, ovp_version, ovp_type, hid, is_on_premise, audience_segment, time_stamp, city
Request-specific	referrer_site, network_id, slot_type_id, ad_request_id, is_not_yume_white_list, publisher_channel_id, content_profile_id, is_pre_fetch_request, service_provider_name
Beacon-specific	slate_id, zero_tracker, twentry_five, fifty, seventry_five, one_hundred, volume percent

Table 1: Selected Feature Table

3.2 Profile Extraction

Yume's raw dataset are based on session records data, each of which correspond to one view behavior of a device. We extracted both beacon and request profile of a device by grouping all record data for that device. The

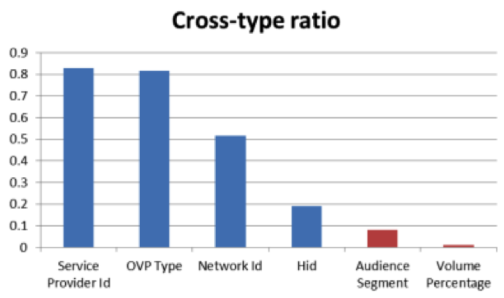

Figure 2: Feature Cross Type Ratio

two profiles are then joined together to form the joined profile.

3.2.1 Beacon and Request Profile

There are two methods employed to generate the device profile: majority-based profile, and majority-set-mixed profile.

Majority-based profile: for categorical attributes(e.g. publisher_id), we use the majority value to represent; for numerical attributes(e.g. view_tracker), we use the frequency.

Majority-set-mixed profile: for most of categorical attributes, we used set to include every possible value in the device profile (e.g. content_video_id, ovp_type etc), except boolean attributes such as is_prefetched.

3.2.2 Joined Profile

After extracting beacon and request profile of each device, we do a full join to form joined profile. Here we excluded outlier devices that have more than 500 records within a week since we believe those devices are unlikely to be a valid human user. At last, we got about 28 million device profiles from California. Thereafter, we focused on grouping those device profiles into users.

3.3 LSH based Candidate Generation

Since we have millions of devices, it would be quite computationally expensive if we compare every pair of devices, thus we will firstly use Locality Sensitive Hashing (LSH) to get candidate pairs that are likely to be from the same user.

3.3.1 Ideal Identical Hash

We convert numerical features to intervals. For categorical data along with intervals, as the string values represent distinct identities, we would use Ideal Identical Hash (IIH) functions which only hash the same value into the same bucket [2]. It is therefore also required that the hash functions that have low collision rate. We tested and compared the collision rate and runtime benchmark of 5 hash functions(i.e. fnv32a, djb2, sdbm, md5, sha256), and eventually chose SHA256 as our hash function.

3.3.2 Convert Columns to Signature

For union-set based attributes in mixed profiles, we need to convert the columns (sets) into signatures before applying the IIH functions. We use min-hashing to achieve this, which preserves the similarity between devices. In implementing min-hashing, we have several permutation vectors for each attribute and concatenate all signatures given by those random permutations to form our final signature. The signature function of column C is defined as the index of the first row that has a 1 in column C:

$$\mathrm{Sig}_i(C) = min(\pi_i(C))$$

3.3.3 Band Construction

We divided 33 features into 5 overlapping bands. By overlapping bands here, it means that we have features that exist in multiple bands, this essentially puts more weight to some features and eliminate a certain amount of false positives in our buckets. If two devices can be hashed into the same bucket for one band, we will consider them as a candidate pair.

After observing the data, we have picked a few features which are assumed to be more correlated with user identification, e.g. household_id, census_DMA and network_id. We should thus assign more weights for those features in both LSH and similarity (discussed in section 3.4.1). In LSH, more weights mean that those features should be included in more than one band.

3.3.4 Agglomerative Clustering Within the Bucket

After LSH, each bucket usually have no more than 10,000 devices. But an $O(N^2)$ pair-pair edge emission is still very expensive. We thus need to further break down the buckets into smaller groups. We used agglomerative clustering within the bucket. Using similarity metric described in section 3.4, we grouped devices with high similarity into smaller clusters. Then we emitted pair-pair edge within those smaller clusters rather than the large bucket.

3.3.5 Missing Data Estimation

As our datasets are very sparse, many of feature values are not properly recorded in our dataset. For example, many feature records have only "NA?"as their value. In the LSH step, if a feature value is missing, we would randomly generate a value out of all possible values of this feature; in the similarity calculation, we would use expectation to represent the similarity score of this feature.

$$E(S_k) = \Pr(X_k = Y_k)$$

3.4 Similarity Function

Similarity (distance) measures are essential to perform the clustering and selecting the candidate pairs. As the device attributes contain both categorical and numerical features, we require different similarity scheme to account for each type respectively, and then fusion the two scores to obtain the overall similarity score.

3.4.1 Categorical Features

The categorical similarity measures assign a similarity value between two data instances X and Y belonging to the data set D as follows [1]:

$$S(X,Y) = \sum_{k=1}^{d} w_k S_k(X_k, Y_k)$$

Where $S_k(X_k,Y_k)$ is the per-attribute similarity between two values for the categorical attribute A_k. The quantity w_k denotes the weight assigned to the attribute A_k.

The weights we assigned for each attribute are based on its correlation with user identification. For example, we assume that a different Household ID is more likely to result in different user than a different placement id, therefore we will assign more weights on Household ID. In the end, we assign the highest weight on household ID and census DMA, less weights on network id, publisher id and service provider id, and least weights to the rest. The exact weights are assigned based on heuristic.

For majority profiling, per-attribute similarity will be measured by overlapping method: if the two device share the same value (i.e. majority-value) , they have value 1 for this attribute. Otherwise, 0. Then the final similarity score is the average of scores of every attribute.

For union-set based attributes of the mixed profiling, per-attribute similarity will be measured by Jaccard Similarity, which is the size of their intersection divided by the size of their union:

$$Sim(X_k, Y_k) = | X_k \cap Y_k | \,/\, | X_k \cup Y_k |$$

3.4.2 Numerical Features

We use cosine similarity to measure the distance of numerical features:

$$Sim(X,Y) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|}$$

The overall similarity would be a weighted average of categorical similarity (for categorical features) and cosine similarity, the weights are assigned by the ratio of number of categorical features over numerical features.

3.5 Weakly Connected Component based User Prediction

LSH in the previous step only yielded a number of candidate pairs, in this step, we needed further connect candidate pairs into groups, which represents all the possible devices that can be assigned to a single user. To merge pairs together, we can view each device as a node and each candidate pair as an edge in a graph, and then we can use Kruskal's Algorithm to obtain the label all weakly connected components (WCC) within the graph. The number of nodes within each WCC represents the number of devices this user has.

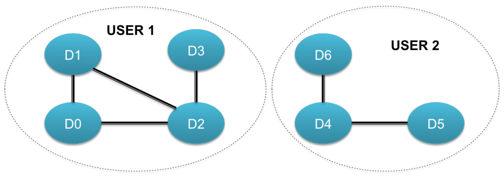

Figure 3: WCC as user

4 Evaluation and Results

Since there is no labeled data available for evaluation, we only proposed three measures to give an sanity-check whether our methods make sense, which involves: devices-per-user distribution, time-location-based error rate estimation, and device-type-based error rate estimation.

4.1 Devices-per-user distribution

In this evaluation method, intuitively the total number of users predicted should be within a reasonable range, and most users should have one to three devices, and only very few users could have more than four or five devices. By drawing the distribution of each type of users, we can roughly tell whether our prediction makes sense. The result of user distribution versus number of devices per user and devices-per-user distribution for both majority based profile and mixed profile are shown in Figure 4 and Figure 5.

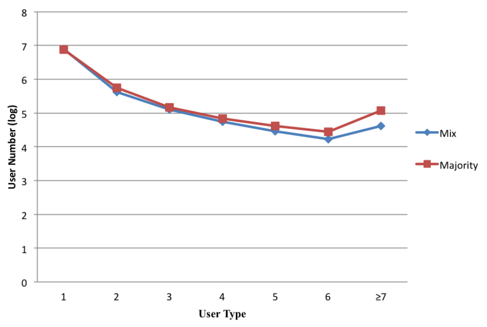

Figure 4: User distribution versus number of devices per user

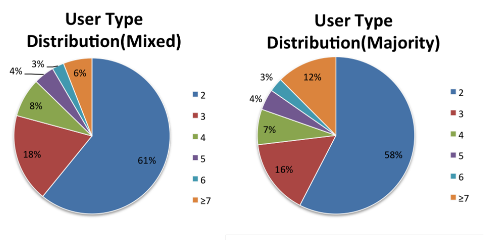

Figure 5: Devices-per-user distribution

The two figures show that the user distribution approximately follow an exponential curve, i.e. the number of users decrease exponentially with the number of devices per user. This curve aligns with the exponentially law in social network studies.

4.2 Time-location-based error rate estimation

Besides the devices-per-user distribution analysis, we also used time and location as evaluation. The intuition is that within a time duration, a user should also be within a reasonable geographic range (e.g. 25 miles / hour). For each device, we could get a list of time-location tuples. Then we compare every pair of time-location tuples from two devices. As long as one pair fails this constraints, we mark this user as a false prediction. Although this evaluation does not help estimate the false negative error rate, it provided a good estimate of the false positive error rate.

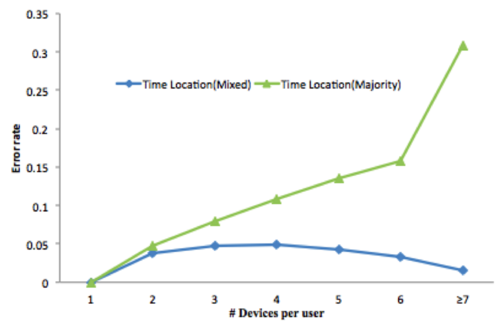

Figure 6: Error rate distribution of # Devices per user

Figure 6 shows the time-location error rate of majority profiling and mixed profiling for each number of devices per user respectively. It is seen that the error rate for mixed is lower than that for majority for each user group. This falls our expectation as the mixed profiles include a larger set of feature values, and thus training parameters in doing LSH and calculating similarities, which will in turn result in lower training error.

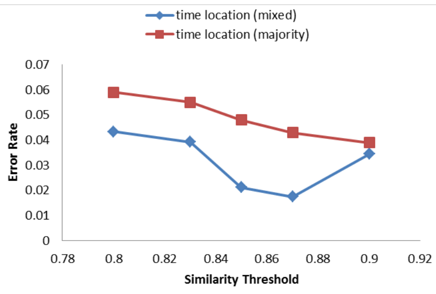

Figure 7: Error rate distribution of threshold

Figure 7 shows the time-location error rate of majority profiling and mixed profiling for each similarity threshold. The higher the threshold, the less number of matching pairs we detect and thus leading to lower error rate. Similar to Figure 6, the error rate of mixed profile is again lower than that of majority profile.

4.3 Device-type-based error rate estimation

Intuitively, it is less likely that a user could have more than two tablets or two smartphones. So using the type of devices (i.e. delivery point), we could also tell whether our predictions are reasonable. As for laptops and smart TV, a unique device could have multiple identifiers, we can only restrict the rules to mobiles and tablets. We define the rule as such: if a user has no less than two devices in the same type, it is a false prediction. Again,this measure only provided a good estimate of the false positive error.

Figure 8 and Figure 9 shows the device-type-based error rate of majority profiling and mixed profiling versus number of devices per user and similarity threshold. Mixed profiling gained much better result in not grouping devices of the same type into one user. Also generally as we increase the threshold, device-type-based error rate tends to decrease, which conforms to our expectation.

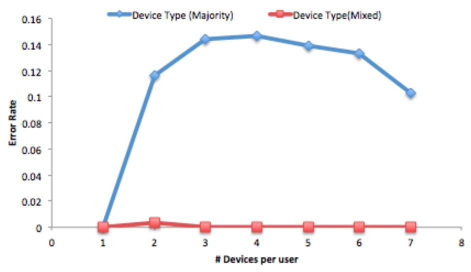

Figure 8: Error rate distribution of # Devices per user

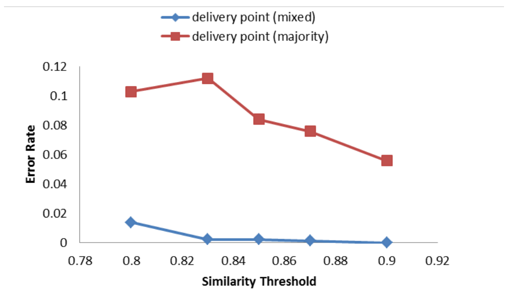

Figure 9: Error rate distribution of threshold

5 Conclusion and Future Work

This report describes a similarity-based approach to cross matching different devices to the same user. In particular two methods are used to formulate device profiles from session records, which are majority based and majority-set mixed based profiling. The locality sensitive hashing are then used to generate all candidate pairs, and weakly connected components are then identified to represent each user group. In the absence of labeled data, we introduced three sanity-check methods to evaluate a good false positive error.

To further improve our work, we should devise a more systematic way to find the optimal weights for each of the feature in band assignment and similarity measure, which is currently only based on heuristic. It is desired to train a sub-set of data with labeled user, and construct a probablistic model which yield the optimal weights with lowest generalization error. However, an accurate and reliable labeling scheme is required here, which can also be used to extend our evaluation model to measure more false negative errors.

We will also investigate a parallel algorithm to identify weakly connected components within the graph, as our current algorithm can only be run on a single cluster, such that it is expensive to compute and the whole graph may not fit in the memory. In order to be more scalable, a parallel version of the algorithm should be investigated.

References

[1] *Boriah, Shyam, Varun Chandola, and Vipin Kumar.* Similarity measures for categorical data: A comparative evaluation, 2008.

[2] BREITINGER, F., AND BAIE, H. Similarity preserving hashing: Eligible properties and a new algorithm mrsh-v2. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 114* (2013, pp 167-182).

[3] CARMAGNOLA, F., AND CENA, F. User identification for cross-system personalisation. *Information Sciences 179.1* (2009), 16–32.

[4] DATTA, ANUPAM, D. S., AND SINHA, A. *Provable de-anonymization of large datasets with sparse dimensions*. No. 229-248. Springer Berlin Heidelberg, 2012.

[5] NARAYANAN, A., AND SHMATIKOV., V. Robust de-anonymization of large sparse datasets. *Security and Privacy, IEEE Symposium* (2008).