CROSS-DEVICE MATCHING

summer 2019 intern project

authors: Matt Dwyer, Hannah Lo, Katie Perkins, and Macy Thompson

zirous®

TABLE OF CONTENTS

zirous®

INTRODUCTION TO ZIROUS

WHO IS ZIROUS?

For over 30 years, Zirous has been providing data-centric solutions to clients in machine learning and artificial intelligence, identity and access management, application integration, and the infrastructure and development needed to support them.

Zirous' machine learning and artificial intelligence team is stacked with world-class machine learning engineers, data analysts, and application developers.

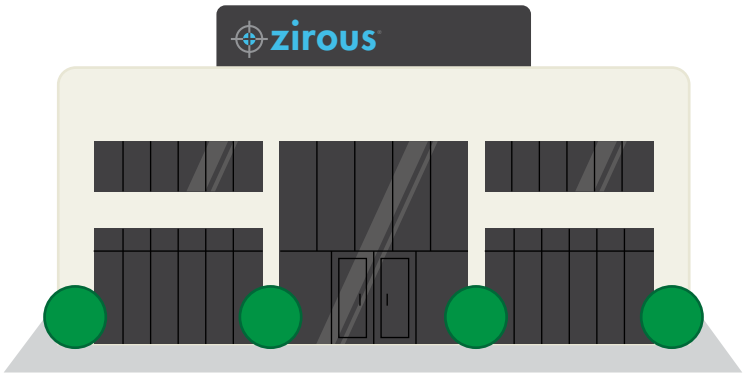
WHY ZIROUS?

Our clients love us for three resounding reasons:

- We provide local and accessible experts that fit company culture.
- We're a partner who understands their business even better than they do.
- We are strategy-focused and not afraid to challenge the status quo.

We know these are true - because we asked! Our clients tell us that we're an essential part of their team and their success because we work **with** them to reach their goals.

All of these things align with our values, which makes this even better news. We're not an in-and-out, one-size-fits-all provider working through as many clients as possible as quickly as possible. We are truly custom solution providers, working through your specific business problems to provide the answers and resources you need to do your job best and boost your bottom line.



zirous®

THE PROBLEM: CROSS-DEVICE MATCHING

The customer journey, from first impression all the way to conversion, is no longer a linear path and therefore complicates correctly crediting a conversion to the appropriate customer touchpoint. Standard attribution methods, which use last point attribution to credit a conversion, fail to recognize that a customer's purchase on a desktop computer should be credited to an Instagram advertisement viewed on their smartphone. Customer journeys like this are becoming increasingly common, with over 40% of online adults beginning an activity on one device and finishing it on another. Attribution models that do not account for user activity across multiple devices fail to tell the real story of a customer's journey and prevent companies from accurately understanding the effectiveness of their marketing efforts. So how does a company fill that gap in knowledge? The solution lies in a concept called cross-device matching. Cross-device matching is the process of connecting multiple devices to the same user. Once a user's devices are known, it makes it easier to analyze their behavior as a whole across multiple platforms, and improve marketing strategy to optimize user experience across all channels.

CROSS-DEVICE MATCHING TERMINOLOGY

DETERMINISTIC & PROBABILISTIC MATCHING

There are two main methods used when implementing cross-device matching: deterministic matching and probabilistic matching. **Deterministic** methods tie multiple devices to one user through personally identifiable information and user IDs. An example of this would be when a user logs into a website on both their phone and their computer. We can then deterministically link those devices to the same user, because the user gave us their information on both devices. Deterministic matching is very accurate, but can only be used if a user explicitly identifies themselves on multiple devices.

Probabilistic matching, on the other hand, utilizes a variety of behavioral and system data, such as IP addresses, location, choice of operating system, and many other features, to conclude how likely it is that the two devices belong to the same user. Based on this likelihood, probable matches can be made. Therefore, probabilistic matching can identify previously overlooked matches, but could also produce some incorrect matches.

"DEVICE" DEFINITION

Without a user identifying themselves, we have no way to tell with absolute certainty that a mobile application and a mobile browser belong to the same physical mobile device. OS-enforced privacy is the main roadblock that prevents this. Similarly, for our purposes, a computer is not a device, but a Safari browser and a Firefox browser on that computer would be two separate devices.

OUR PROCESS

OVERVIEW

1. Collect device data.
2. Determine if the collected device data is from an existing device or a new device.
3. Compare the data from two devices and utilize a device-matching machine learning model to predict the probability of the two devices belonging to the same user.
4. Based on the predicted probabilities, dense connections between devices are determined, representing the set of devices a user has.

DATA COLLECTION

We collected data on our users' activity via an Android application and a script on the Zirous website. The goal of these techniques was to obtain as many pieces of information about our users on each device as possible, in hopes that these device profiles would enable us to compare devices to ultimately conclude which devices belonged to the same user.

Android applications, despite containing privileges that must be granted by the user, allowed for more extensive data collection to occur. On the other hand, computer browser data collection had few, if any, restrictions, but provided less data for us to collect. Browser data also could be misleading at times. Some browsers -- such as Google Chrome -- intentionally give inaccurate information if they predict that you are attempting to collect data to track a user. While not overly common, this inaccurate information made it very difficult to detect browsers in incognito mode, or the drive upon which they were running. On the contrary, mobile browsers provided the most detailed and extensive data to collect, even going so far as to provide the software version and device model of each device that visited the website.

The application and website scripts generated log files for us to collect and organize. The data was organized into "packets" of data and was indexed to determine the event type of that packet (battery, location, browser, geofence, cellular, IP, etc.). We also appended a unique identifier to each mobile application session, called the "zID", and a universally unique identifier on browser sessions, called the "page_load_id". Finally, timestamps were included on all data stored, as collection times often varied form storage times, causing an issue with any time based analytics.

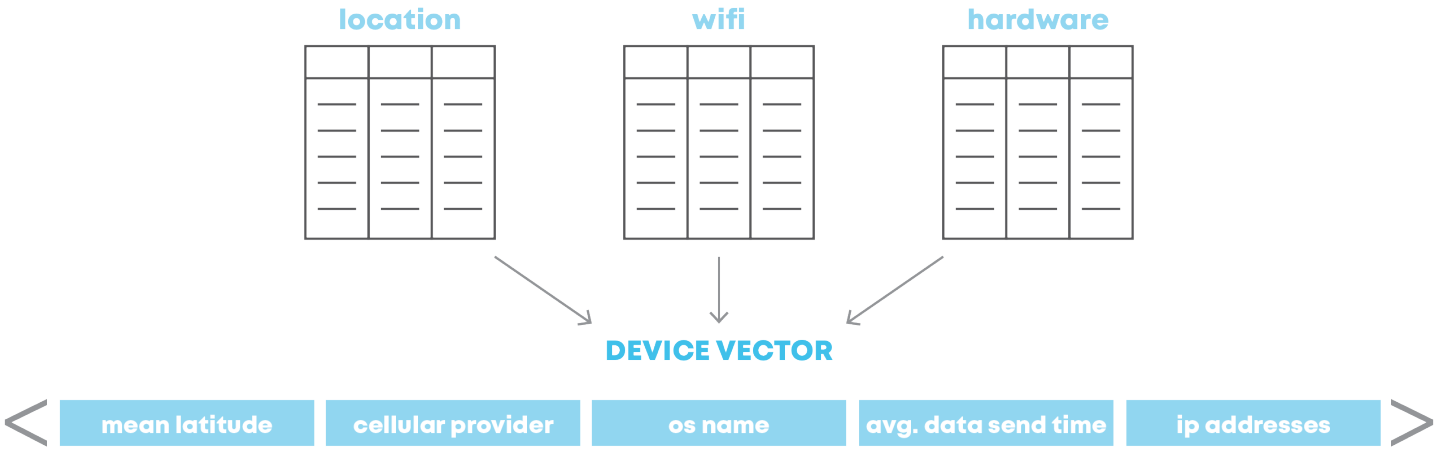
Utilizing streaming technology, the data packets were micro-batched then routed to our general storage location within an AWS S3 bucket. Upon storage, a stream processor function (AWS Lambda) took each line of data, labeled and indexed it, and inserted it into a relational database in respective tables. Lambda functions also allowed for us to iterate through adjustments to our on-the-fly data transformations as our process evolved. It then would send each data packet to our data pipeline step function, which is where our matching process begins.

MATCHING

Before we jump into the matching process, we need to define two important terms that we use. An accepted technique to find a solution to this problem is to use something called a **device signature** to help with the matching process. The idea is to collect a variety of data features that are not unique individually, but when combined become very unique as a whole. Ideally, the signature is unique enough that if an identical signature comes in, there is a high probability that it belongs to the same device. We implemented similar concepts through what we called **device vectors** and **similarity vectors** to perform two different types of matching: matching incoming data to a device, and then devices to users.

DEVICE VECTORS: ASSOCIATING INCOMING DATA TO A DEVICE

Before we could try to identify what user the data belonged to, we had to identify the data as a device. The device vector associates the incoming data event types to an existing device vector if a match was found, or creates a new device vector if the data seems to be coming from a new device:



To determine if the data belongs to an existing device vector, the data was first sent through our deterministic matching function. Some of the identifiers we used for deterministic matching are in the table below:

identifier	description
login / Google single sign on	Requires a user to log in from a device, which allows for direct association between this incoming data and a known user.
GUID	A "globally unique identifier," which is unique to an application installation instance. Requires user permission to use for tracking.
ad ID	A globally unique identifier provided to a single device by Google. Requires user permission to use for tracking.
phone number	Requires user permission for application to read cellular data.
subscriber ID	A unique cellular subscriber ID, for example, the international mobile subscriber identity for a GSM phone. Requires user permission for application to read cellular data.
specific cookies	Some cookies contain ids such Google Advertising IDs or MailChimp IDs that can be extracted and used to match deterministically.

If a deterministic match could not be made, the data was then sent through our probabilistic matching process. Due to there being no deterministic match, we know any included unique identifiers would not be helpful at this stage. Instead, the goal was to find a set of features that, when in combination with each other, create a signature that is (almost) unique to a single device– in other words, an existing device vector. Using a mix of rules-based matching and sending device vectors through a machine learning model, a probability would be calculated to predict the likelihood that the packet of data is from an existing device.

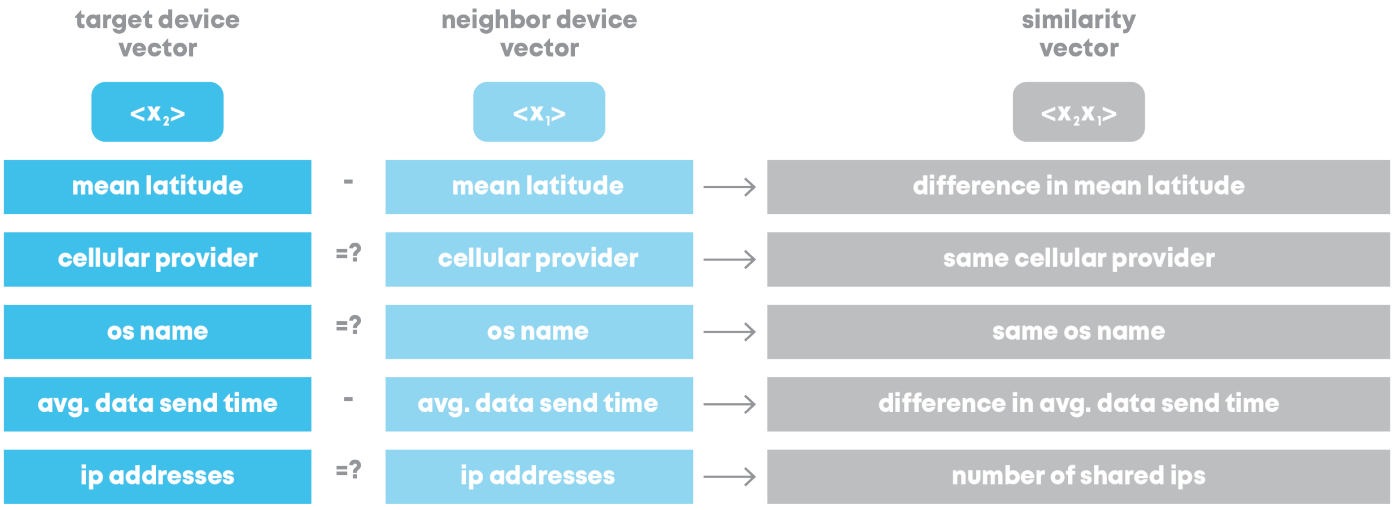
When a deterministic or probabilistic match was found, the device vector was updated with the most recent data values. If no existing device vector seemed to match, a new device vector was created and populated with the packets of data.

SIMILARITY VECTORS: COMPARING THE SIMILARITY OF TWO DEVICES

In the same way as associating data to a device, first we wanted to consider unique identifiers that would allow us to deterministically associate devices to a user. Some of the deterministic features we considered were:

login / Google single sign on	Requires a user to log in from both devices, which allows for direct association of these devices to a known user.
background browser collection in app	A discrete web browser that is opened in the Android application that executes the browser collection activities and deterministically links the application ID to the browser information.
email campaigns	An email campaign, like MailChimp, allows you to generate an ID that is unique to each email account. When the user clicks the link to visit the website from a campaign email, their activity is associated with that ID. This ID is consistent across all devices so long as the email link is clicked from the same email account.

It's impossible to expect deterministic data for all customer and prospect digital interactions. Many won't sign in or identify themselves, and will visit from several different devices. In order to probabilistically match devices, we needed to create our similarity vector to determine how similar two device vectors were. The similarity vector stores the differences and similarities between the data of two specific device vectors:



We found that both the application data and the browser data we were collecting both included the following features, among others:

- Number of shared IP addresses
- Difference in median/mean latitude day/night
- Difference in median/mean longitude day/night
- Ads enabled/disabled
- OS name
- OS version
- Device type (mobile or browser)
- Timezone
- Language settings
- Time of day with most activity
- Difference in average time of day that data is sent
- RAM
- CPU Cores

This allowed us to make comparisons across different device types (such as our mobile app and computer or mobile browsers), store them in similarity vectors, and then analyze the similarity vectors to determine if those two devices belong to the same user.

ANALYSIS: ASSOCIATING DEVICES TO A USER

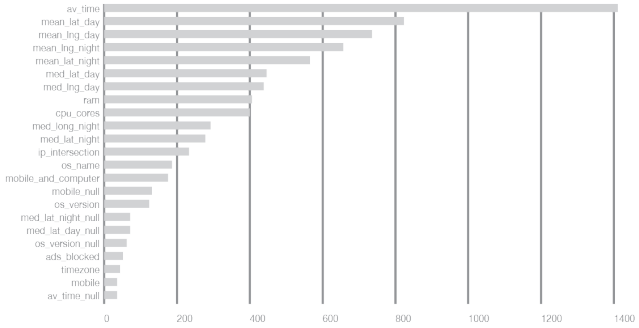
machine learning

Our similarity vectors were sent in batches to our machine learning model, accessed through a Sagemaker endpoint.

Before we could do this, however, we had to train our ML model. The training data for our ML model was collected from our users when they tested our Google login on both the mobile application and web browsers through our website. We used the device identifiers associated with each Google login username to build our ground truth data. We then created our training data, which consisted of the features from the similarity vectors and a label of “yes” or “no”, representing whether the two devices comprising the similarity vector belonged to the same user or not. Since our data was highly imbalanced with negative examples (comparing each device concluded no match significantly more often than concluding a match), we chose to use the gradient boosting algorithm called XGBoost for its demonstrated performance with low signal data. Using a balance of true positive rate (recall) and overall accuracy, our final evaluations showed our model at a 95% accuracy and 83% recall.

After the model made its predictions, it returned a set of weights representing the probability that the two devices used to create the similarity vector belong to the same user. XGBoost also determines which features were most important in making a final decision on the weight. The top three most important features in the similarity vector were the following: difference in average time of day that data is sent from a device, difference in mean latitude during the day, and difference mean longitude during the day.

top segment of decision tree generated by XGBoost

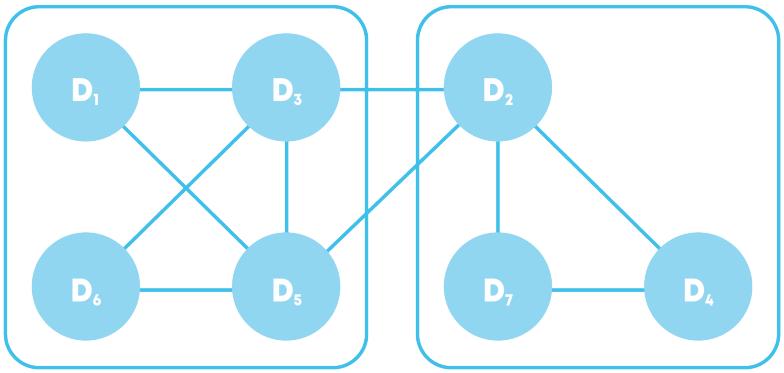


feature importance

graph database

The weight predicted by our machine learning model was only predicting the likelihood that two devices belonged to the same user. We needed to take that one step further to identify all of the devices that belonged to the same user. To do this, we utilized a graph database.

We set up our device graph to be an undirected, weighted graph, with nodes representing devices and edge weights representing the probability that two devices (or nodes) belong to the same user. The higher the probability, the more dense the graph. Each set of densely connected devices represents a single user, and the probabilities within the graph are updated every time our model produces new probabilities. Below we can see an example of what a graph and its densely connected devices would look like.



It is also worth mentioning that it could be more accurate to use some form of a rolling average or smoothing function for updating the probabilities of the graph, rather than only using the most recent prediction from the machine learning model. A rolling average would reduce the abrupt impact one-time events could have, and maintain the density of the graph.

APPLICATIONS FOR BUSINESS

Once a user creates an account and identify themselves, their activities and preferences can be associated to the individual, but what about before that account is made? What was their journey before they made the decision to make a purchase, when they were virtually anonymous? And how do you link the unknown and known profiles together? Regardless of your industry or what entices your users to grant you access to their data, tracking users from their first visit to their first payment and beyond is crucial to building a unified view of a customer.

Cross-device matching enables companies to reliably track their customers across multiple devices, even before account creation. This allows a business to do many things, such as identify initial roadblocks for potential customers, personalize new user interactions to drive customer conversions, and group customers by similar behaviors for targeted advertising. Behavioral data, along with technical information about their devices, can be used to create a holistic, omni-channel view of the customer.

The most interesting problem that cross-device matching tackles is the issue of linking an anonymous customer to an identified customer. As an anonymous customer visits a website or uses an application, their activity on different devices is probabilistically linked to a single, unidentified person. As their interactions continue and more information is collected, that anonymous profile becomes more easily recognizable. This recognizability allows for previously collected data to be connected to future data once an account is created, tying all activities of one individual together.

This new field will allow businesses to answer questions that have previously been difficult to quantitate, and can help to optimize marketing strategies to ensure that customers convert. By calculating similarities between devices to determine individual user tendencies, a personalized experience can be created for customers beginning with their first interaction, one that spans across all avenues of communication.

CONTACT

machine learning & artificial intelligence lead
Madison Lang
madison.lang@zirous.com
(515) 974-5567

vice president of sales
Mike Spear
mike.spear@zirous.com
(515) 974-5522

headquarters
1503 42nd Street
West Des Moines, IA 50266

/company/zirous

@zirous

/zirousinc

zirous.com

zirous®