IP Geolocation through Reverse DNS

OVIDIU DAN*, Lehigh University, USA
VAIBHAV PARIKH, Microsoft Bing, USA
BRIAN D. DAVISON, Lehigh University, USA

IP Geolocation databases are widely used in online services to map end-user IP addresses to their geographical location. However, they use proprietary geolocation methods and in some cases they have poor accuracy. We propose a systematic approach to use reverse DNS hostnames for geolocating IP addresses, with a focus on end-user IP addresses as opposed to router IPs. Our method is designed to be combined with other geolocation data sources. We cast the task as a machine learning problem where, for a given hostname, we first generate a list of potential location candidates, and then we classify each hostname and candidate pair using a binary classifier to determine which location candidates are plausible. Finally, we rank the remaining candidates by confidence (class probability) and break ties by population count. We evaluate our approach against three state-of-the-art academic baselines and two state-of-the-art commercial IP geolocation databases. We show that our work significantly outperforms the academic baselines, and is complementary and competitive with commercial databases. To aid reproducibility, we open source our entire approach and make it available to the academic community.

CCS Concepts: • **Information systems → Location based services**; • **Networks → Location based services**; *Network measurement*; Public Internet; • **Social and professional topics → Geographic characteristics**.

Additional Key Words and Phrases: IP geolocation, hostname geolocation, IPv4, IPv6, geographic targeting, geotargeting, local search, contextual relevance, geographic personalization, reverse DNS

ACM Reference Format:
Ovidiu Dan, Vaibhav Parikh, and Brian D. Davison. 2022. IP Geolocation through Reverse DNS. *ACM Trans. Internet Technol.* 22, 1, Article 17 (February 2022), 30 pages. https://doi.org/10.1145/3457611

1 INTRODUCTION

IP Geolocation databases map IP addresses to their corresponding geographical locations. They are used to find the approximate location of an IP address at the city level. Records in these databases typically contain IP ranges along with their physical location. These databases are vital to a variety of online services when the exact location of a user is not available. Table 1 lists a few examples of such records. For instance, the second example in the table maps a /24 subnet (256 IPs) to Hengyang, a city in China. While some users opt-in to share their exact coordinates to online services through mobile devices with global positioning sensors, others decline or use devices without such features. IP geolocation is therefore a valuable source of information on user location.

*Also with Microsoft Bing.

Authors' addresses: Ovidiu Dan, Lehigh University, Bethlehem, PA, USA, 18015, ovd209@cse.lehigh.edu; Vaibhav Parikh, Microsoft Bing, Redmond, WA, USA, 98052, vparikh@microsoft.com; Brian D. Davison, Lehigh University, Bethlehem, PA, USA, 18015, davison@cse.lehigh.edu.

1533-5399/2022/2-ART17 $15.00
https://doi.org/10.1145/3457611

Table 1. Example of entries from an IP Geolocation database

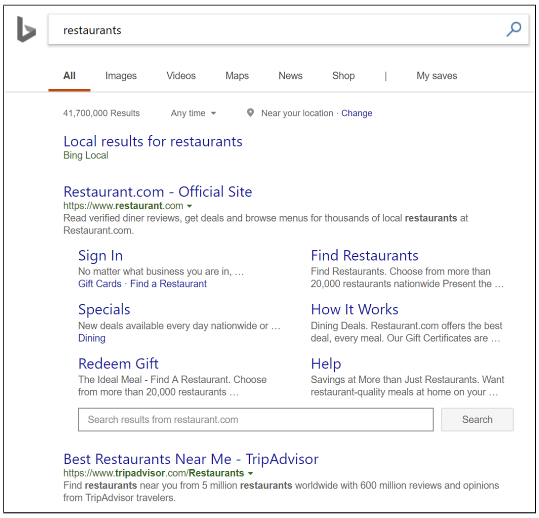
StartIP	EndIP	Country	Region	City
1.0.16.0	1.0.16.255	JP	Tokyo	Tokyo
124.228.150.0	124.228.150.255	CN	Hunan	Hengyang
131.107.147.0	131.107.147.255	US	Washington	Redmond

A practical application of IP geolocation is **personalized local search results** in the context of web search engines. When issuing queries such as *weather* and *zoo near me*, users are implicitly requesting local results even if they do not specify a city in the query. Figure 1 demonstrates the striking difference in results for the query "restaurants" when the user location is unknown, compared to when it is known. The top image shows the case when an IP lookup in a geolocation database fails to find a user's location, while the bottom image shows the case when the location is found and it is correct. The generic results lead to nationwide websites where the user has to requery for more specific restaurants in their area, while the personalized results directly list restaurants tailored to a specific location. Previous work has shown that personalizing results to a user's location leads to increased user satisfaction and conversely that missing location information leads to user dissatisfaction [4, 14, 37]. IP geolocation databases are also used in many other applications, including:

- **Content Personalization**: Personalize content based on the location of the user, for example for displaying local weather information or local news, or displaying content in the native language of the user [4, 29, 58].
- **Content Delivery Networks**: Direct users to the closest datacenter to decrease latency and increase network throughput [16, 32].
- **Credit card fraud protection**: Use the location of users making online payment transactions as one of the inputs in their credit card fraud detection algorithms [2, 5].
- **Online advertising**: Personalize ads based on users' location such as showing local deals [38].
- **Law enforcement**: Fight botnets and cybercrime by tracking the location of IP addresses, and aid in detecting unauthorized logins [7, 34, 40, 55].
- **Geographic content licensing**: Restrict audio or video content delivery to licensed geographic regions [39, 61].
- **E-commerce**: Automatically determine taxes and shipping while displaying prices in online stores [57].

Companies such as MaxMind, Neustar IP Intelligence, and IP2Location provide state-of-the-art commercial IP geolocation databases. They combine multiple IP location sources, including WHOIS lookups, network latency information, network topology information, reverse DNS, as well as direct contracts with Internet Service Providers [43]. Related work has shown that while they have high coverage, commercial databases are sometimes inaccurate or are missing location information for some IP ranges [17, 26, 47, 54]. Our own past work has demonstrated that the city-level accuracy of geolocation databases is less than 70% in ten major countries [14]. Furthermore, the methods they use to derive their databases from these raw data sources are proprietary and therefore impossible to properly reproduce and expand upon in an academic settings. **In contrast, here we focus on describing and evaluating a single IP geolocation approach in detail, while also making the source code available to other researchers.**

Our work focuses on extracting location information from reverse DNS hostnames assigned to IP addresses. Extracting location information from hostnames has many potential advantages



restaurants

All Images Videos Maps News Shop | My saves

41,700,000 Results Any time Near your location · Change

Local results for restaurants
Bing Local

Restaurant.com - Official Site
https://www.**restaurant**.com
Read verified diner reviews, get deals and browse menus for thousands of local **restaurants** at Restaurant.com.

Sign In
No matter what business you are in, …
Gift Cards · Find a Restaurant

Find Restaurants
Find Restaurants. Choose from more than 20,000 restaurants nationwide Present the …

Specials
New deals available every day nationwide or …
Dining

How It Works
Dining Deals. Restaurant.com offers the best deal, every meal. Our Gift Certificates are …

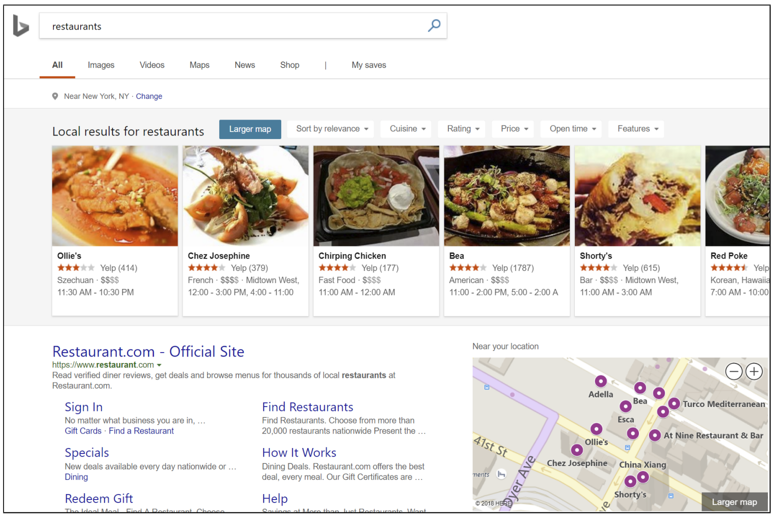
Redeem Gift
The Ideal Meal - Find A Restaurant. Choose from more than 20,000 restaurants …

Help
Savings at More than Just Restaurants. Want restaurant-quality meals at home on your …

Search results from restaurant.com Search

Best Restaurants Near Me - TripAdvisor
https://www.**tripadvisor**.com/**Restaurants**
Find **restaurants** near you from 5 million **restaurants** worldwide with 600 million reviews and opinions from TripAdvisor travelers.

(a) User location is unknown



restaurants

All Images Videos Maps News Shop | My saves

Near New York, NY · Change

Local results for restaurants Larger map Sort by relevance Cuisine Rating Price Open time Features

Ollie's
★★★☆☆ Yelp (414)
Szechuan · $$$$
11:30 AM - 10:30 PM

Chez Josephine
★★★★☆ Yelp (379)
French · $$$$ · Midtown West,
12:00 - 3:00 PM, 4:00 - 11:00

Chirping Chicken
★★★★☆ Yelp (177)
Fast Food · $$$$
11:00 AM - 12:00 AM

Bea
★★★★☆ Yelp (1787)
American · $$$$
11:00 - 2:00 PM, 5:00 - 2:00 A

Shorty's
★★★★☆ Yelp (615)
Bar · $$$$ · Midtown West,
11:00 AM - 3:00 AM

Red Poke
★★★★★ Yelp
Korean, Hawaiia
7:00 AM - 10:00

Restaurant.com - Official Site
https://www.**restaurant**.com
Read verified diner reviews, get deals and browse menus for thousands of local **restaurants** at Restaurant.com.

Sign In
No matter what business you are in, …
Gift Cards · Find a Restaurant

Find Restaurants
Find Restaurants. Choose from more than 20,000 restaurants nationwide Present the …

Specials
New deals available every day nationwide or …
Dining

How It Works
Dining Deals. Restaurant.com offers the best deal, every meal. Our Gift Certificates are …

Redeem Gift

Help

(b) User location is detected as New York City

Fig. 1. Effect of missing location information on search engine local search personalization. The top image displays the results for the query "restaurants" when the location of the user is unknown, while the bottom image displays the personalized experience for a user located in New York City.

including high coverage and accuracy, when compared to other approaches such as network delay geolocation. Reverse DNS hostnames can be periodically collected in a short amount of time by performing a reverse DNS lookup for every address in the IP space. Figure 2 exemplifies the information that can be parsed from reverse DNS hostnames. In this case we can derive both the location and connection characteristics from the hostname of an IP address. A person reading the name of the hostname can reasonably expect that it references *Wallingford*, a town in Connecticut, USA.

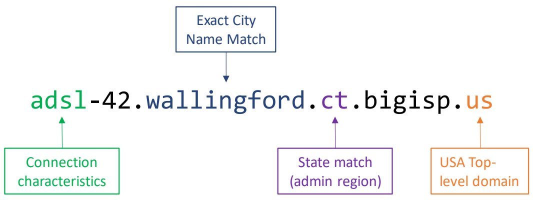
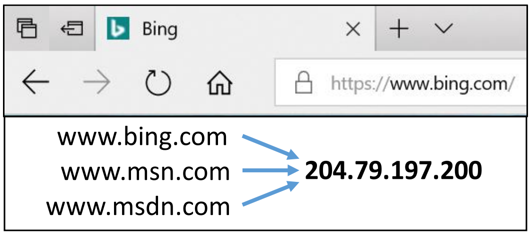
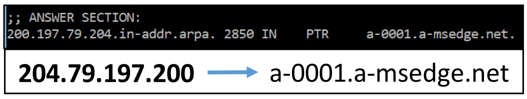

Exact City Name Match
adsl-42.wallingford.ct.bigisp.us
Connection characteristics
State match (admin region)
USA Top-level domain

Fig. 2. Example of information that can be extracted from reverse DNS hostnames.

Reverse DNS is the opposite of forward DNS. Figure 3 contains examples of both forward and reverse DNS resolution. Forward DNS starts from a hostname such as `www.bing.com` and resolves to zero, one, or more IP addresses [42]. Note that multiple hostnames can map to the same IP. Conversely, reverse DNS lookups start from an IP address and typically returns zero or one hostnames [19]. As in our example, the reverse DNS hostname does not need to be the same as the forward DNS hostname. While forward DNS lookups are used by Internet users to get to websites, reverse DNS hostnames are typically used to name and describe the underlying physical infrastructure that makes up the Internet.



(a) Forward DNS example. Multiple hostnames can be mapped to the same IP address.



```
;; ANSWER SECTION:
200.197.79.204.in-addr.arpa. 2850 IN    PTR     a-0001.a-msedge.net.
```

(b) Reverse DNS example. An IP address typically has zero or one reverse DNS hostnames.

Fig. 3. Example of the difference between forward DNS and reverse DNS resolving. The former starts from a hostname and maps to one or more IPs; the latter starts from an IP and maps to zero or one reverse DNS hostnames.

We propose a systematic approach for using reverse DNS hostnames to geolocate IP addresses. We focus on end-user IP addresses, which we show make up the vast majority of IP addresses, and which are most useful in IP geolocation for online services. Our approach is designed to be combined with other data sources when compiling an IP geolocation database. **Given a reverse DNS hostname, our task is to determine a list of geographic locations at the city level that could reasonably match**. This task poses multiple challenges. First, the naming schemes of Internet Service Providers are often ad-hoc and do not always contain the full names or common abbreviations of cities. For example, the `drr01.cral.id.frontiernet.net` hostname is located in *Coeur D'Alene, Idaho*. Determining that the `cral` substring maps to this location is difficult even for a human. Second, many cities around the world have ambiguous names. Take for instance *Vancouver, Canada* and *Vancouver, USA*. A hostname which only contains the substring *vancouver* is not specific enough to determine a single location correctly. Even unambiguous city names can become ambiguous when abbreviations are used instead of their full names. Does `nwmd` refer to ***N**e**w** Rich**m**on**d***, *WI* or to ***New** **M**aryland*, *NB*, or to neither of them? Third, sometimes hostnames contain conflicting locations. For example, it is difficult to determine if `sur01.tacoma.wa.seattle.comcast.net` is located in *Seattle, WA*, *Tacoma, WA*, or maybe even ***Su**mne**r***, *WA*. More specifically, our contributions are:

(1) **We study how reverse DNS hostnames evolve over the course of five years**. Among other findings, we determine the usage of reverse DNS hostnames is increasing in time, and that there are 1.25 billion reverse hostnames across the entire IPv4 space, of which hundreds of millions appear to contain location hints.
(2) **We present a machine learning approach for extracting locations from hostnames**. We cast the task as a machine learning problem where for a given hostname, we split the hostname into its constituent terms, we generate a list of location candidates, then we classify each hostname and candidate pair using a binary classifier to determine which candidates are plausible. Finally, we rank the remaining candidates by confidence, and we break the ties by population.
(3) **We evaluate our approach against state-of-the-art baselines**. Using a large ground truth set, we evaluate our approach against three academic baselines and two commercial IP geolocation databases. We show that our method significantly outperforms academic baselines. We also show that the academic baselines contain incorrect rules which impact their performance. Finally, we demonstrate that our approach is both competitive and complementary to commercial geolocation baselines, which shows that our method can help improve their accuracy.
(4) **We release our data and code as open source**. To help the academic community reproduce our results, we open source the components of our reverse DNS geolocation system, including the hostname splitter, sampling strategy, features, and the classifier itself. The code is available at **github.com/microsoft/ReverseDNSGeolocation**.

Before we delve into the related work in this area, we would like to make a note of how we use the terms *domain* and *subdomain* in this paper. Figure 4 shows an example of how we name different parts of the hostname `router.nwestnet.net` in our work. According to RFC 1034, *a domain is a subdomain of another domain if it is contained within that domain* [42]. Therefore technically `[net]`, `[nwestnet.net]`, and `[router.nwestnet.net]` are all subdomains of the root domain. Furthermore, they can also each be domains. However, in this paper we use these terms in the more colloquial sense where the domain is a name (for example `comcast.net` and `bing.com`) that can be registered through a registrar, the subdomain *fragment* is the part of the hostname to the left of the domain, and the subdomain is equivalent to the entire hostname. In Section 5.1 we will further introduce the term *logical domain* for special cases where for reverse DNS geolocation purposes we use a subdomain as the domain. Finally, top-level domains such as `net`, `com`, `org`, `us` are the domains in the DNS tree that are immediately under the common root domain, which is the parent of all domains in the DNS tree.

2 RELATED WORK

We divide IP geolocation research in two broad categories, based on the methods they use: **network delay and topology** approaches use ping, traceroute, and BGP network structure information; **Internet data mining** approaches use diverse information mined from the Internet, including web page content, WHOIS databases, reverse DNS, and social graphs.

The majority of IP geolocation research relies on active network delay measurements to locate addresses. Early work on IP geolocation by Padmanabhan and Subramanian discusses

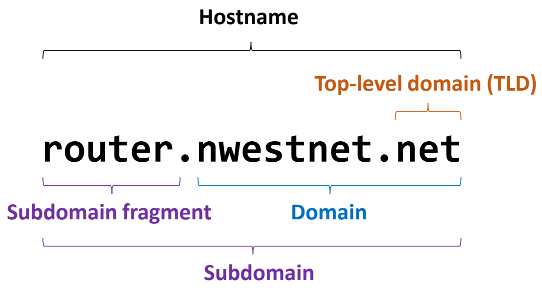


Fig. 4. Example of naming different parts of a hostname in this paper. See text for details.

GeoPing [46], which sends ICMP packets from geographically distributed landmark servers to the target IP. It then assign the target IP the location of the closest landmark server in terms of latency. *CBG* [27] goes further by creating circles on the surface of the earth around each landmark server, where they calculate the radius of each circle based on their measured network delay. It then uses multilateration to infer the location of the target IP at the intersection of these circles. *GeoCluster*, also proposed by Padmanabhan and Subramanian [46], combines BGP routing information with sparse IPs of known locations to assign geographical locations to whole address prefixes. *TBG* [35] uses traceroute from landmark servers to the IP target and perform global optimization to find the location of both landmarks and targets. Youn et al. [67] develop a statistical method for IP geolocation based on applying kernel density estimation to delay measurements. More recently, Ciavarrini et al. [12] presented a framework to understand how the position of landmarks and their distribution affect localization performance. Multiple systems such as Octant [66], Alidade [11], or HLOC [53] combine delay measurement methods with other data sources such as reverse DNS and WHOIS information.

Network delay and topology methods have significant limitations. First, all such methods require access to nodes spread throughout the globe to perform measurements. Second, geolocating a large number of IP addresses using network measurements can run into scalability issues, as each target IP address or range requires separate measurements. The ZMap project from the University of Michigan can scan the entire IPv4 address space using a gigabit connection [18]. However, performing useful network delay measurements would require a significant number of such machines distributed around the world, and attempting to perform traceroutes would require running this probe step once for every hop distance. Third, not all networks allow ICMP pings or fully disclose their network topology. Fourth, routes on the Internet do not necessarily map to geographic distances. Fifth, the ground truth data for work in this area is usually limited to a few tens of IP addresses, typically located in the United States. For example, *GeoPing* and *GeoCluster* are evaluated on only 256 target IP addresses, all located at universities in the United States, *CBG* is evaluated on only 95 IP addresses in the U.S. and 42 addresses in Western Europe, *TBG* only targets IP addresses located at U.S. universities, etc. Sixth, previously reported mean and median errors of tens to hundreds of kilometers show that these methods cannot be used for practical applications at the city granularity. For instance, *GeoPing* has an error distance of error distance of 150 kilometers at the 25th percentile, and *CBG* has median error of 100 kilometers for some datasets. Seventh, previous work has shown that the accuracy of these approaches depends on both the proximity of the landmarks to the target address and on the density of landmarks [12].

Our work addresses several of these limitations. First, using reverse DNS hostnames for geolocation does not require any network delay measurements. Reverse DNS hostnames can be obtained much faster than performing active delay measurements, by querying DNS servers. Second, our ground truth dataset is several orders of magnitude larger than the ones used in previous work and it spans the entire planet. Third, our approach can be performed offline and is scalable. Fourth, our results have much better median error distance than most previous research.

Internet data mining approaches use diverse information mined from Internet, such as web page content, IP whois information, reverse DNS hostnames, social networks, etc. Guo et al. [28] extract locations mentioned in web pages and assign the locations to the IPs which host the content. Using an IP geolocation database as ground truth they report an agreement on the city level for 87% of the IPs. This work has two obvious problems: first, an IP geolocation database with unknown accuracy is used as ground truth; and second, the method focuses on geolocating IP addresses of servers, not of end users. Endo and Sadok [20] propose using *whois* information. Unfortunately, the evaluation section lacks a comparison against ground truth. Wang et al. [63] combine the *CBG* approach with extracting the location of web servers from the web pages that they host. Using the web servers as landmarks they are able to achieve good results, with a median error of 0.69 kilometers for the best data set, which contains only 88 IP addresses. The approach partially suffers from the same scalability limitations as network delay based methods, as it requires access to extensive worldwide Internet architecture to run. The results also depend on the density of servers in a geographical area that can be used as landmarks. In recent years the migration of local web servers to cloud services has accelerated, which makes it more and more difficult to find local landmarks [63].

Backstrom et al. [3] propose an interesting approach which relies on a user's social graph to determine their location. They derive the location of target users based on the locations of friends. Using self-reported location as ground truth they show an improvement over an unnamed IP geolocation database. For an error distance of less than 25 km, the amount of correctly classified IPs increases from 57.2% for the baseline to 67.5% for the proposed method. The authors state that this method works so long as an individual has a sufficient number of friends whose location is known, preferably more than 16. This approach yields a median error distance of 590 km on a test dataset of 2,830 IPs.

In this work we propose extracting IP locations from their reverse DNS hostnames, which we also classify as an Internet data mining approach. *GeoTrack* [46], proposed by Padmanabhan and Subramanian, is one of the earliest reverse DNS geolocation approaches. They create manual rules to determine locations of hostnames in the United States using city names, airport codes, and country codes. They then combine this approach with traceroutes to estimate the location of a target IP. In contrast, our machine learning approach does not require manual rules and it achieves a median error of only 17.5 km on a test set of 1.6 million worldwide IPs.

Undns is the most well-known and widely used reverse DNS geolocation approach [56]. Similarly to *GeoTrack*, it consists of manual rules. The rules are expressed as regular expressions at the domain level. The extracted slots in the rules are then matched to manually generated lists of locations. For the rule `([A-Z]{3,4})[0-9]?.verizon-gni.net` *undns* matches the hostname `PHIL.verizon-gni.net`. The rule looks for subdomain fragments with 3 or 4 uppercase letters, followed by an optional numeric digit. A domain specific location dictionary is then used to match the extracted slot `PHIL` to *Philadelphia, PA*. The obvious disadvantage of this approach is that each domain requires manually generated and thus potentially error prone rules. *undns* does not extract any location, if there are no rules for a domain, or the rules are incomplete. The hostname `PHLAPA-LCR-08.verizon-gni.net` does not match any rule, although it is likely also located in *Philadephia, PA*. This is an example where undns would fail. Also, its accuracy is unknown because the paper does not present a specific IP geolocation evaluation. In comparison, our approach is much more scalable, since it does not require human generated rules. It also handles unique situations better, since it considers the terms of each hostname individually, without requiring domain specific training. Our system is also easier to keep up to date by just running the automated retraining process. **In Section 6.3 we show that our approach significantly outperforms undns.** Several geolocation and network topology papers use undns as-is to draw conclusions or perform experiments [24, 44, 47, 64]. However, we demonstrate that *undns* results suffer significantly due to catch-all rules.

DRoP, another state-of-the-art reverse DNS based approach, aims to geolocate hostnames using automatically generated rules generated by finding patterns across all the hostname terms of a domain [33]. For example, it may find that for the domain `cogentco.com`, the second term from the right often contains airport codes. These rules are then validated using network delay information. DRoP places 99% of IPs in 6 test domains within 10 kilometers of their actual location. However, it uses network delay measurements which would require significant worldwide computing resources and run time to scale to the entire IP space. In contrast, our proposal does not require delay measurements as it only parses hostnames. Also, DRoP is designed for and evaluated against router hostnames, when most geolocation applications need to geolocate end user residential IP addresses. Our proposal works on hostnames of both end user and router IP addresses. Furthermore, DRoP uses a method of splitting hostnames that is rudimentary compared to ours and can miss some location hints. Our approach generates multiple interpretations of how a hostname can be split into parts. For example, our approach can split on the transition between letters and numbers, not only on the dotted parts. Finally, DRoP was only tested on a handful of ISP domains that were known beforehand to have consistent reverse DNS naming rules with location hints. In comparison, our approach can work on any unseen ISP hostnames even without a consistent naming scheme. We show here that our system significantly outperforms DRoP in error distance.

DDec [21] combines undns and DRoP rules by giving precedence to undns and using DRoP as fallback. Unfortunately, we demonstrate in Section 6.3 that DDec and its constituent parts perform poorly on worldwide ISP domains due to incorrect and catch-all rules. **To perform a fair comparison, we compare our approach against DDec (undns + DRoP) on domains that the baselines specifically support, although our approach can scale to unseen domains and to hostnames with ad-hoc naming that do not follow a consistent naming scheme.**

HLOC, which is more recent work by Scheitle et al. [53], is similar to DRoP in that it extracts location hints from reverse DNS hostnames, and it validates them using network delay measurements. However, it uses the location hints directly to construct a candidate location list to be verified, whereas DRoP also aims to output specific hostname parsing rules. This work has several problems. First, Scheitle et al. do not properly evaluate HLOC against a ground truth set. Instead, they determine agreement with commercial databases and with DRoP. These results in themselves do not tell us the accuracy of HLOC. Second, they consider any location hint on a radius of 100 kilometers around city centers to be located in that city, which is not necessarily true in high density regions with many cities and towns close to each other. Third, as with DRoP, they specifically target only router IPs and filter out any residential addresses. Fourth, they ignore any location hints for places that have less than 100,000 inhabitants. In the United States, only ≈300 cities have a population greater than 100,000, out of more than 35,000 cities and towns in the country. Fifth, due to several filtering steps, this approach could only extract locations for 4.7% of IPs in their router dataset.

We previously published a preliminary version of this work that focused on the engineering side of developing a distributed version of our approach [15]. This paper greatly expands on our previous work. First, this work focuses on describing the features used for reverse DNS geolocation in detail, whereas our previous work primarily describes the engineering challenges in adapting our work to be distributed. In our previous work we performed experiments on a cluster of 2,000 machines and demonstrated that our distributed implementation is more than 150 times faster than a single-machine version. In Section 5 of this work we focus on describing the features and reverse DNS geolocation classifier. Second, here we present a detailed motivation for our work, as well as a greatly expanded discussion on background information and related work, in Sections 1 and 2. Third, Section 4 of this paper contains an an analysis of the Rapid7 Reverse DNS dataset across multiple years. Fourth, in the same section we also determine the percentage of infrastructure IP addresses (routers as opposed to end-user IPs) by intersecting the Rapid7 dataset with the CAIDA Macroscopic Internet Topology Data Kit (ITDK) dataset [8]. Fifth, in Section 5.3 we present an improved classifier which outperforms our previous work. Sixth, we compute and discuss more evaluation metrics, such as RMSE. We also compare our approach against commercial geolocation services across more hostname domains, in Section 6.4. Seventh, in Section 6.3 we expound on examples where the academic baselines we compare against fail to parse hostnames. Eighth, in Section 7 we discuss the privacy implications of our work. Finally, along with this paper we make our work available as open source to allow other researchers to reproduce our results, as described in Section 8.

3 DATASETS

This section contains descriptions of the datasets we use throughout this article for experiments, training and testing:

- **Ground truth**: We train and evaluate our approach using a ground truth set of 67 million IP addresses with known world-wide location. **To the best of our knowledge, this dataset is the largest and most diverse ever used in IP geolocation literature**. We compiled the ground truth set in March 2018 by randomly sampling search impressions from Bing query logs. We describe the characteristics of this dataset in detail in Section 6.1, and we discuss privacy considerations in Section 7.
- **GeoNames** is a free database with geographical information about the entire world at different levels of granularity, such as streets, neighborhoods, cities, countries, and continents. This continuously updated database can be accessed either through a Web API or by downloading data dumps [65]. The March 2018 snapshot we used contains information on 11.5 million geographic features from all countries in the world. We specifically used the following subsets available separately for download:
  - **Cities 1000** consists of information on all cities in the world with a population of at least 1,000, including original names, ASCII names, alternate names, coordinates, and the codes of administrative divisions at several levels of granularity. For example it states that Paris has 2.1 million inhabitants, it is part of the administrative region with code 11, and it is the capital of France. It also provides latitude and longitude for the city center, as well as 105 alternate names in multiple languages.
  - **Alternate Names** contains more alternate names for some cities such as abbreviations, colloquial names, and historic names. More importantly, it also contains **airport codes** issued by *IATA*, *ICAO*, and *FAAC*, which are travel organizations. For Paris this dataset lists the closest IATA airport code to be PAR, which is the common code for all airports in the Paris region.
  - **Admin 1 Codes** is comprised of the codes and names of first-level administrative regions. Continuing with the Paris example, we find that code 11 refers to Île-de-France, which is one of 18 regions in France and it contains the Paris metro area.
  - **Country Info** contains general information about countries, including the population, currency, postal code format, languages, and the Internet top-level domain (TLD). For Paris, France the TLD for websites is `.fr`.

- **CLLI** is an abbreviation for Common Language Location Identifier. These codes are used by the North American telecommunications industry to designate names of locations and functions of telecommunications equipment. While historically only used by the Bell Telephone companies, they were more recently adopted by other companies as well. Multiple codes can map to the same location. For example, all the following codes map to *Chicago, Illinois*: `chcgil`, `chchil`, `chciil`, `chcjil`, and `chclil`. Note that the codes cannot necessarily be derived from the name of the city. This database is available from multiple sources. We acquired a May 2017 snapshot from TelcoData [60] for a token amount.
- **UN/LOCODE**, which stands for United Nations Code for Trade and Transport Locations, is a worldwide geographic coding scheme developed and maintained by the UN. It assigns codes to locations used in trade and transport, such as rail yards, sea ports, and airports. The code assigned to Paris, France is `FRPAR` and the functions listed for this location are: `port`, `rail`, `road`, and `postal`. The codes are obtained by concatenating two-letter ISO 3166-1 alpha-2 country codes with three characters that more specifically describe the location of physical infrastructure used in trading. This dataset is updated twice a year and it is available for free on the United Nations Economic Commission for Europe website [22]. We used the December 2017 release, which was the latest available dataset when we performed our evaluation.
- **Public Suffix List**, which is maintained by the Mozilla Foundation, is a list of domain suffixes under which Internet users can directly register names [23]. Some examples include `cloudapp.net`, `gov.uk`, and `konsulat.gov.pl`. The list can be used to extract the subdomain fragment from a hostname. This dataset is updated daily and it is free. Please see Section 5.1 for more details.
- **Rapid7 Reverse DNS** consists of reverse DNS hostnames of **the entire IPv4 address space**. The dataset is provided for free to researchers by Rapid7 Labs and it is updated weekly. The archive contains snapshots going back to 2013 [50]. We discuss the dataset in detail in the next section.

4 REVERSE DNS

Performing forward DNS lookups converts valid hostnames such as `www.bing.com` into IP addresses, while performing reverse DNS lookups works in the other direction. Since the forward and reverse DNS lookups are defined by different DNS records, they do not need to have the same hostname. Reverse hostnames are more likely used to name the underlying networking infrastructure, as opposed to forward hostnames that are used to name websites or other online services. Both forward and reverse DNS are described in RFC1034 [42], an Internet standard published by the Internet Engineering Task Force.

Reverse DNS lookups are achieved by querying PTR and CNAME records. To perform a reverse lookup of the IPv4 address `204.79.197.200` we query the PTR record for the hostname `200.197.79.204.in-addr.arpa`. We obtain this hostname by reversing the four octets of the IP address, such that `204.79.197.200` becomes `200.197.79.204`, then we append the `in-addr.arpa` domain. The DNS tree is walked backwards, so first the nameserver for `in-addr.arpa` is resolved, then the one for `204.in-addr.arpa`, etc. This structure assumes that IP addresses are allocated by Internet registries to ISPs in blocks of 256 IP addresses or more, since the lookup eventually reaches `197.79.204.in-addr.arpa`. While this was historically true, with the introduction of classless inter-domain routing addresses started being allocated in smaller blocks. To address the problem of reverse DNS hostnames for smaller blocks, RFC2317 [19] proposed using CNAME records to further divide each block if needed.

IPv6 addresses also have reverse DNS hostnames. The only difference is that the records are under the `ip6.arpa` domain. **While we evaluate our approach on IPv4, all methods described in this article can be equally applied to IPv6 addresses as well.**

To determine the viability of using reverse DNS hostnames for geolocation, we began our investigation by studying a dataset provided by Rapid7 Labs [50]. This dataset, which is updated once a week, is available for free and it consists of reverse DNS hostnames of the entire IPv4 address space. Rapid7 obtains the records by performing IPv4 PTR lookups over the entire address space as described above, except for ranges that are blacklisted or private. The archive contains snapshots going back to 2013 [49]. The preliminary investigation in this section is based on a snapshot taken in January 2018, while in the rest of the article we use a more recent dataset from March 2018.

We first discuss the coverage of reverse DNS hostnames in the IPv4 address space, which consists of all 32-bit numbers. This limits the possible address space to 2^{32} (4.3 billion) addresses. The number of usable IP addresses is actually only 3.7 billion, since the Internet Engineering Task force and the Internet Assigned Numbers Authority designated some IP ranges as special-use or private [1]. Since not all IP addresses have a reverse DNS hostname, we parsed the Rapid7 dataset to find the actual reverse DNS coverage. We found that 1.25 billion addresses have a reverse DNS hostname. This finding shows that while they have massive coverage, reverse DNS hostnames cannot be used alone in determining location information for the entire IP space. This partial data source must therefore be combined with other data to obtain a complete geolocation database.

We then quantified how many of the hostnames are valid, as the DNS records are unrestricted strings. We parsed each hostname and rejected the ones that did not respect Internet host naming rules [6, 30]. For example, we ignored hostnames which contain spaces. We also rejected hostnames that did not have a valid suffix as defined by the *Public Suffix List*, which is a list of valid domain suffixes previously described in Section 3. This left us with 1.24 billion hostnames, of which 1.15 billion were distinct. Our findings are summarized in Table 2, which shows that 33.4% of usable IP addresses have a valid reverse DNS hostname, and 31.1% are distinct. Considering that not all IPv4 addresses are yet allocated, the actual percentage is likely higher.

This work focuses on geolocating end-user IP addresses, therefore it is useful to know what percentage of IPs with a valid reverse DNS hostname are instead used by Internet infrastructure such as backbone routers. To make this determination we intersected the Rapid7 Reverse DNS dataset with the router dataset from CAIDA's Macroscopic Internet Topology Data Kit (ITDK) project [8], from the same time frame. ITDK contains data about connectivity and routing gathered from a large cross-section of the global Internet. One of the files in the dataset contains a list of all the routers detected on the Internet, along with the IP addresses of their ports. These two datasets have 35.6 million IP addresses in common. This corresponds to about 1% of usable IP addresses and 3.1% of distinct reverse DNS hostnames. The result shows that, as expected, the vast majority of addresses are used by end-users.

Next, we set out to determine if reverse DNS hostnames are a valuable source of geolocation information. We searched for exact city names and airport codes in the hostnames, using the *Cities 1000* and the *Alternate Names* dataset, respectively. We found that 163.7 million hostnames could contain exact city names, and 272.9 million hostnames could contain airport codes. This approach represents an upper-bound of the number of hostnames that could contain exact city names or airport codes. The results contain true positives such as `sur01.seattle.wa.seattle.comcast.net` in *Seattle, Washington* and `inovea5.gs.par.ivision.fr` in *Paris, France*. However, this naive approach also matches false negatives such as `node-j.pool-1-0.dynamic.totbb.net` which is not in *Pool, UK* and `mobile.bigredgroup.net.au` which is not in *Mobile, Alabama*. Nevertheless, the results summarized in Table 2 show that there are potentially hundreds of millions of hostnames that could contain geographic information, using just these two features alone. We conclude that while the results are promising, a more sophisticated approach could achieve higher coverage and accuracy.

To further familiarize ourselves with hostname naming conventions, we extracted the top subdomain fragment components of the largest 10 domains in the Rapid7 dataset. To obtain the components we split each subdomain fragment on the dotted components, and then we further split the components on dashes and on the transitions between numbers and letters. For example, hostname `soc-l.wht2.ocn.ne.jp` has a subdomain fragment `soc-l.wht2` and a domain `ocn.ne.jp`. We split the subdomain fragment into `soc`, `l`, `wht`. We then manually labeled the components that we found to reasonably correspond to geographic locations. We also cross-checked our findings with commercial databases.

Table 3 shows the results. We observe that only 4 out of the top 10 domains contain indicators of geographic location. However, those that use geographic encodings do so extensively. We find that service providers use various naming conventions across different networks and within a single network. For instance, the hostnames under the *sbcglobal.net* domain owned by *AT&T* make use of abbreviations such as `pltn` to refer to *Pleasanton, CA*. But they also use combinations of city abbreviations with State names such as `chcgil` to refer to *Chicago, Illinois*. Our findings are in line with work by Chabarek and Barford [10], who found that all eight of the providers they studied used multiple naming schemes.

We also studied the distribution of top level domains such as `.com` and `.fr` in the *Rapid7* dataset to determine if country-specific domains can be used as location hints. We observed that most hostnames contain a `.net` domain at 33.2%, followed by `.com` with only 17.2%. This is the opposite of forward DNS, where `.com` is more popular. The difference is due to Internet Service Providers preferring to use `.net` TLD domains for hostnames that describe the underlying physical architecture of their *network*. After removing the `.com`, `.net`, `.edu`, and `.mil` domains which together make up 51.6% of valid hostnames, we are left with approximately 600 million hostnames, the vast majority of which are country-specific. We found very few novelty TLDs used in reverse DNS hostnames. We conclude that the corresponding country of a reverse DNS domain could be a useful hint in geolocation.

Finally, we compared snapshots of the dataset, each collected in the month of January of years 2014 to 2018, inclusive. Our goal was to determine how the characteristics of the hostnames change in time. For each IP in the snapshot we compared the hostname values in consecutive years. Table 4 shows a summary of the results. We found that a maximum of 14.3% of hostnames changed year over year. This number includes the cases where one side of the comparison contained a hostname but the other side was empty due to the DNS query returning an empty hostname, or

Table 2. Usage of reverse DNS hostnames across the entire IPv4 space. More than 1.24 billion IPv4 addresses contain valid reverse DNS hostnames.

Set name	Size	% of usable	% of distinct
Total IPv4 space	**4.3 B**		
↪ Reserved IP addresses	0.6 B		
↪ Usable IP addresses	3.7 B		
↪ IPs with reverse DNS hostnames	1.25 B	33.7%	
↪ Valid reverse DNS hostnames	1.24 B	33.4%	
↪ Distinct DNS hostnames	1.15 B	31.1%	
↪ Exact city match (naïve)	0.16 B	4.4%	14.1%
↪ Airport code match (naïve)	0.27 B	7.4%	23.5%
↪ Internet Infrastructure IPs	0.04 B	1.0%	3.1%

Table 3. Top hostname components of the largest 10 domains that have reverse DNS hostnames. We manually highlighted locations with underlined blue. The percentages in the *valid* and *usable* columns are computed based on rows 3 and 5 of Table 2.

Domain	Count	% of valid	Top hostname components sorted in descending order by how often they appear in all hostnames of this domain
comcast.net	50.0M	4.0%	c, hsd, hsd1, m, **ca**, **pa**, **fl**, **il**, **ma**, **ga**, a, **co**, **mi**, d, f, **wa**, b, e, **va**, **nj**, **or**, **md**, **tx**, **chlm**, **chic**, **phil**, **tn**, **in**, npls, dd, **atlt**, **sjos**, **denv**
bbtec.net	37.2M	3.0%	softbank, biz
rr.com	31.1M	2.5%	res, cpe, mta, **socal**, biz, rrcs, **nyc**, neo, **nc**, **wi**, kya, **columbus**, **cinci**, **carolina**, **tx**, **central**, **twcny**, **nycap**, **west**, **sw**, **rochester**
myvzw.com	29.6M	2.4%	sub, qarestr
sbcglobal.net	28.4M	2.3%	lightspeed, adsl, dsl, **irvnca**, **hstntx**, **rcsntx**, **cicril**, **sntcca**, **tukrga**, **miamfl**, **pltn**, **pltn13**, **stlsmo**, **livnmi**, **bcvloh**, **frokca**, **chcgil**
t-ipconnect.de	24.5M	2.0%	dip, dip0, p, b, e, a, f, d, c, pd, fc, fd, fe, ff, de, dd, dc, df, ee, bb, bd, bc, ae, ac, aa, ab, af, ad, ba, bf, ea, eb, be, ec, fa, ed, fb, ef, db, da, ca, cf
telecomitalia.it	19.4M	1.6%	host, static, business, b, r, retail, dynamic, host156, host15, host94, host61, host127, host232, host112, host95, host72, host107, host220
ge.com	16.7M	1.4%	static, n, n003, n003-000-000-000, n129, n144, n144-220-000-000, n129-201-000-000, n129-202-000-000, n165-156-000-000m n165, n192
ocn.ne.jp	16.2M	1.3%	p, ipngn, **tokyo**, **osaka**, ipbf, **marunouchi**, ipbfp, omed, omed01, **kanagawa**, **hodogaya**, **aichi**, **osakachuo**, **saitama**, **hokkaido**
spcsdns.net	16.0M	1.3%	pools, static

due to the request failing because of network failures during data collection. We then performed a similar comparison, this time counting only the cases where both sides of the comparison contained non-empty hostnames. Here we found that a maximum of 2.2% hostnames change over the years, if both of the values are present. To understand why there is such a large discrepancy between these two findings we also determined the number of hosts that were gained or lost between the years. By hostnames gained we mean that in the older year a hostname was missing, while in the subsequent year it was present, and by hostnames lost we mean the opposite. The results show that yearly more hostnames are gained than lost. However, the number of hostnames gained every year has steadily declined from 108.5 million in the first pair, to 79.6 million in the last pair. Conversely, the number of hostnames lost has increased from 44 million to 78.4 million, respectively. Although there are still more hostnames gained than lost yearly, this gap is narrowing. We also found that 64.8% of them remained the same and were non-empty across all five years. This shows that a large fraction of hostnames don't change year-over-year.

Table 4. Reverse DNS hostname changes across 5 years

Change / Year Pair	2014 →2015	2015 →2016	2016 →2017	2017 →2018
Hostnames changed - one hostname in the pair can be empty	**174.0M** (14.3%)	**154.3M** (12.3%)	**157.5M** (12.3%)	**166.6M** (12.9%)
Hostnames changed - both hostnames in the pair are non-empty	**26.4M** (2.2%)	**20.7M** (1.6%)	**14.2M** (1.1%)	**16.7M** (1.3%)
Hostnames gained in the second year of the pair	**108.5M** (9.7%)	**89.3M** (7.6%)	**81.0M** (6.7%)	**79.6M** (6.5%)
Hostnames lost in the second year of the pair	**44.0M** (3.9%)	**49.4M** (4.2%)	**69.6M** (5.7%)	**78.4M** (6.5%)

In summary, we determined that 1.15 billion IP addresses have reverse DNS hostnames, many of which contain exact city or airport code matches. We also determined that reverse DNS hostnames are becoming more prevalent year over year.

5 APPROACH

We cast the problem of extracting locations from reverse DNS hostnames as a machine learning problem. Given a hostname, our proposed approach splits the hostname into components, finds a preliminary list of location candidates, generates primary and secondary features for each candidate, then classifies each potential location using a binary classifier into two classes - *likely* or *unlikely*, also giving each candidate a confidence score.

For instance, for the hostname `ce-salmor0w03w.cpe.or.portland.bigisp.net`, our technique first splits the hostname into substrings such as *ce, salmor, w, cpe, or*, etc. It then finds a list of potential location candidates whose names match each of these substrings, then unions them into a final candidate list for the entire hostname. In this example, our approach considers tens of potential location candidates, including *Portland, UK* and *Salmoral, Spain*. It then runs a binary classifier on **each** of these location candidates. The inputs to the classifier are the original hostname and one of the location candidates. The outputs are a binary label which signifies if the location candidate is plausible, and a confidence value. We discard the candidates where the output label is false, then we sort the remaining candidates by confidence, breaking ties using city population. For our example, it ranks *Salem, Oregon* and *Portland, Oregon* as the most likely candidates. Since our approach can output multiple plausible location candidates, it can optionally be further combined with other data sources to further rank the candidate list.

5.1 Splitting hostnames

Drawing from our experience of manually labeling hostnames with likely locations in Section 4, we implemented multiple heuristics for splitting hostnames into their constituent components.

First, we apply the *ToUnicode* algorithm described in RFC 3490 [45] to convert International Domain Names (IDN) to Unicode. For example, the hostname `xn--0rsod70av79j.xn--j6w193g` gets converted to 夏威夷舞.香港. The reason we perform this translation is that international hostnames are stored as ASCII strings using Punycode transcription. This allows us to perform location lookups in the hostname using foreign languages.

Second, we separate the *logical subdomain fragment* from the *logical domain* and the public suffix, using the list provided by the Mozilla Foundation previously described in Section 3. These public suffixes are a superset of normal TLDs because they also contain entire domains under which users can create subdomains. For example, the list contains the pseudo-TLD `azurewebsites.net` since users of Azure cloud services can register their own subdomains under this name.

With the aid of Figure 4 we previously explained the naming we use in this paper for the terms *domain*, *subdomain*, *subdomain fragment*. Here we expand upon that naming by introducing the terms *logical domain* and *logical subdomain fragment*. When a public suffix is the same as a TLD, then the extracted *logical domain* is the same as the colloquial definition of domain. So for instance for hostname `router.nwestnet.net` both the TLD and the publix suffix are the same (`net`), so both the domain and logical domain are the same. However, that is not always the case. Take for instance the example in Figure 5. Here the TLD is not the same as the public suffix. The domain is `ne.jp`, but since this is also a public suffix `ne.jp`, we extract a logical domain of `nttpc.ne.jp`.

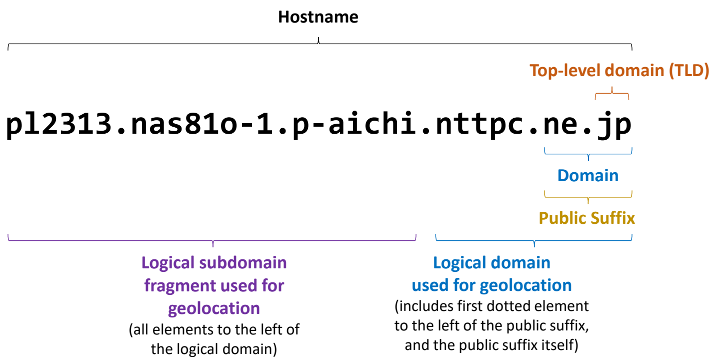

Hostname
Top-level domain (TLD)
pl2313.nas81o-1.p-aichi.nttpc.ne.jp
Domain
Public Suffix
Logical subdomain fragment used for geolocation
(all elements to the left of the logical domain)
Logical domain used for geolocation
(includes first dotted element to the left of the public suffix, and the public suffix itself)

Fig. 5. Example of hostname where the domain is not the same as the logical domain we use for geolocation, because the TLD is not the same as the public suffix. Please see text for details.

In this second step we also extract the TLD. For `pl2313.nas81o-1.p-aichi.nttpc.ne.jp` we extract `nttpc.ne.jp` as the logical domain because `ne.jp` is a public suffix, we extract as the logical subdomain fragment `pl2313.nas81o-1.p-aichi`, and finally we extract `.jp` as the TLD. We perform this step of separating the subdomain fragment from the logical domain and of extracting the TLD from the hostname in order to focus our parsing efforts on the subdomain fragment and on the TLD. The rest of the public suffix rarely contains location hints.

Third, we split the extracted subdomain fragment at three levels of aggregation: on the dotted elements, on hyphens within the dotted elements, and on the transitions between letters and numbers within the hyphenated elements, saving the results at each level. Figure 6 contains a specific example represented intuitively as a tree structure. The bottom three levels of the tree correspond to the three levels of aggregation. As a last step, we trim the leaf nodes. We remove any leaf node consisting solely of numbers. We also remove common terms related to network connection characteristics, such as `static`, `dsl`, `fiber`, and `nas`. We obtained them by counting the top extracted leaf nodes in the training set and manually selecting the ones which are unrelated to geolocation but clearly related to the underlying network infrastructure. The list is available in the source code we are publishing along with this article.

5.2 Features

Starting from the results of the hostname splitter, we find potential location candidates and compute primary and secondary features for **each** candidate. Figure 7 shows a concrete example. Primary features can be derived directly from a hostname. These features are matched using a single contiguous string which indicates a location at city-level granularity. Primary feature generation and candidate selection happen at the same time. Secondary features are generated in the context of a specific location candidate. These features require the context of a primary candidate to match. In our example two location candidates and their primary features are first selected based on the term `roch` in the hostname. Then we compute secondary features separately for each candidate. In the context of *Rochester, Minnesota*, we match the `mn` term as a secondary feature that captures administrative regions for this candidate.

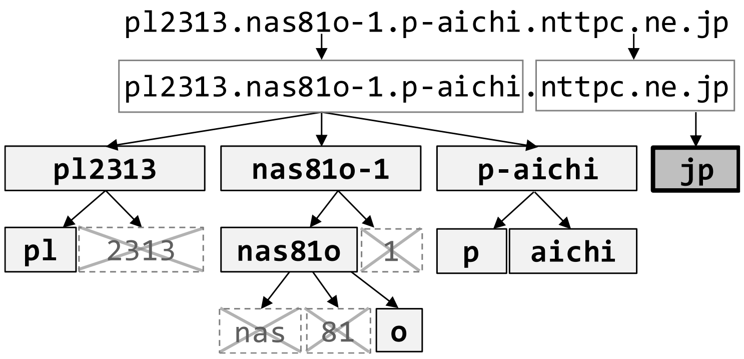


Fig. 6. Hostname Splitter example. The leaf nodes marked with X were removed by the pruning process. The remaining leaf nodes are the output of the splitter.

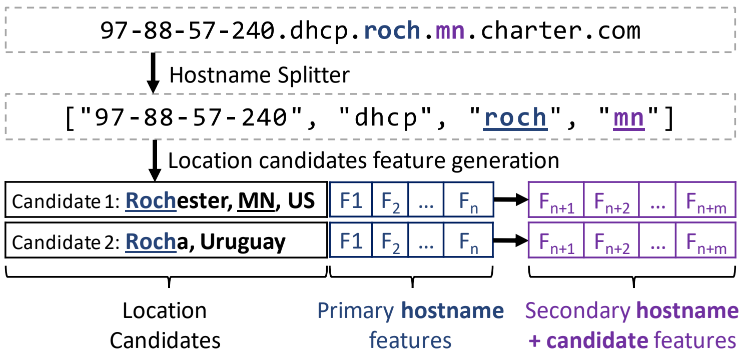

97-88-57-240.dhcp.roch.mn.charter.com
Hostname Splitter
["97-88-57-240", "dhcp", "roch", "mn"]
Location candidates feature generation
Candidate 1: Rochester, MN, US
Candidate 2: Rocha, Uruguay
F1 F2 ... Fn
Fn+1 Fn+2 ... Fn+m
Location Candidates
Primary hostname features
Secondary hostname + candidate features

Fig. 7. Feature Matching and Generation. We precomputed a mapping from terms such as roch to a list of candidates and their primary features

Primary features are based on the *GeoNames*, *UN/LOCODE*, and *CLLI* datasets described in Section 3. From *GeoNames* we use the *Cities 1000*, *Alternate Names*, and *Admin 1 Codes* subsets. Each of the features derived from these datasets consist are mappings of strings to locations. For example, the case insensitive string of each CLLI code maps to a single location. Another example is city names, which in case of ambiguity can match to multiple locations. Table 5 lists an example for each feature category. We describe these feature categories in more detail later in this section. Each of the categories in the table actually maps to three specific features: *IsMatch*, *Population*, and *MatchedLettersCount*. The *IsMatch* feature is a boolean which indicates if the feature matched the current hostname and current location candidate. The *Population* feature contains the population of the current location, if *IsMatch* is *true*. We use population as a proxy for the importance of a city candidate. Finally, *MatchedLettersCount* contains the number of characters which matched. As the number of characters in common between a hostname and a location increases, it could mean a higher confidence match. For instance, if the hostname contains the letters `seattle` and the current location candidate is *Seattle, Washington*, then the *CityName-MatchedLettersCount* Feature would have a value of seven.

While most feature categories in Table 5 are self-explanatory, we describe them here briefly. The *City Name* category matches entire names of cities. *Alternate names* matches translations and colloquial names of locations. *Abbreviations* are based on the first letters of cities with longer names,

Table 5. Primary Feature Categories

Category	Example	Location
City Name	p907072-li-mobac01.**osaka**.ocn.ne.jp	Osaka, Japan
Alternate names	178235248188.**warszawa**.vectranet.pl	Warsaw, Poland
Abbreviations	cpe-68-173-83-248.**nyc**.res.rr.com	New York City, NY, USA
City + Admin1	**torontoon**-rta-1.inhouse.compuserve.com	Toronto, ON, Canada
City + Country	er1-ge-7-1.**londonuk**5.savvis.net	London, United Kingdom
No Vowels Name	static-50-47-60-130.**sttl**.wa.frontiernet.net	Seattle, WA, USA
First Letters	97-90-205-107.dhcp.**losa**.ca.charter.com	Los Angeles, CA, USA
Airport Code	62.80.122.50.**fra**.de.eunx.net	Frankfurt, Germany
CLLI Code	99-166-111-251.**tukrga**.sbcglobal.net	Tucker, GA, USA
UN/LOCODE	16.151.88.129.**krsel**19d.kor.hp.com	Korea, Seoul
Host Patterns	**atoulon**-651-1-29-109.abo.wanadoo.fr	Toulon, France

such as `sf` for *San Francisco*. The *City + Admin1* category consists of concatenations of city and administrative regions, such as `seattlewa`. Similarly, *City + Country* matches combinations of city and country names.

The intent of the *No Vowel Name* feature is to match city names without vowels. It allows partial matches using the first 3 or more letters of the names. For example, this allows matching `gnvl` to *Greenville, SC* and `rvsd` to *Riverside, CA*. Furthermore, based on our observations we further extended this feature with more complex variations. We select the first and last letters of each word in the name, even if the letters are vowels. We then generate combinations of letters from this list, in order. Examples matched by this variation include `oxfr` for *Oxford, MA*, and `ftmy` for *Fort Meyers, FL*.

The *First Letters* features use the first consecutive letters of locations. The *Airport Code* category spans airport codes from travel organizations. *CLLI* and *UN/LOCODE* codes match telecommunications and transportation codes of locations.

Finally, *Host Patterns* attempts to capture rules not encompassed by the other features. For example, `wanadoo.fr` often prepends *a* to location names, as in **`a`**`puteaux` instead of `puteaux` for *Puteaux, France*. However, our hostname splitter does not split terms on consecutive alphabet letters, so it will extract the term as `aputeaux`, which will not match any location. Using the training data we aim to find patterns in the terms of hostnames, separately for each ISP logical domain. First, we group the data per domain. Then we perform pattern mining inside the hostnames of each domain. For a given domain, we intersect all known reverse DNS hostnames in that domain with the data we use for training. This yields a subset of training hostnames in that domain for which we know the correct ground truth location. For each hostname in this subset we then split the subdomain part into its components as previously described in Section 5.1. Suppose for example that we know training hostname `static-32-213-114-101.wlfr.ct.frontiernet.net` is located in Wallingford, Connecticut. We extract subdomain components `wlfr` and `ct`, while ignoring the numbers and the term `static`, which was removed by the pruning process discussed in Section 5.1. Component `wlfr` is in position 1 if we read the remaining components from left-to-right (*ltr*) and position 2 if we read the components from right-to-left (*rtl*). Conversely, component `ct` is in position 1 if we read the subdomain fragments from right-to-left and in position 2 from left-to-right. For the domain `frontiernet.net` we can then extract patterns such as *(wlfr|ltr-1), (wlfr|rtl-2), (ct|rtl-1), (ct|ltr-2), (wlfr|ltr-1,ct|rtl-1), (wlfr|ltr-1, ct|ltr-2)*, etc., all mapping to the city

Table 6. Secondary Features Categories

Category	**Candidate**	**Match Example**
Admin1	Johnstown, PA, USA	246-119.**jst**.**pa**.atlanticbb.net
First Letters Admin 1	Fort Huachuca, AZ, USA	**frth**-bw-noc.**ariz**.army.mil
Country	Paris, France	ci77.**paris**12eme.**fr**.psi.net
Country TLD	Barcelona, Spain	**barcelona**.fib.upc.**es**

Wallingford, Connecticut. By aggregating these rules across all the ground truth training examples in the `frontiernet.net` domain and by using association rule mining [31], we can determine which combinations of patterns have enough support to be mapped to specific locations. In our example, for the domain `frontiernet.net` we can then determine that rule (wlfr|ltr-1, ct|ltr-2) has enough support to be mapped to Wallingford, but rule (ct|ltr-2) is too generic and does not.

To optimize run-time complexity we propose precomputing primary features and location candidates. We start from all known geographic locations and go backwards to generate all possible hostname terms that could match these locations. For example, the *CLLI* dataset contains seven codes for *New York*, including *nyccny* and *nycpny*. We know that the *CLLI* feature can only match *New York* if one of these strings is present as a term in the hostname. Therefore, we can pre-populate a map where the keys are these codes, and the values are the corresponding location (New York), along with precomputed features such as how many letters would match. As we precompute more types of features, we merge them into the same existing map. Given any hostname term, this map will in the end contain all possible locations that match that term, along with all the features from all the categories that match.

For the *CLLI* feature category example, we iterate over each location and generate the substrings that could match based on the corresponding *CLLI* codes. For each location and substring combination we then generate and store the features for this feature category. Finally, we merge the candidate into *Features*, which is a multi-dimensional map where the first level contains all the substrings that could match any candidate feature, the second level contains all the location candidates that could match this substring, and the third level is the precomputed features for this particular substring and location candidate combination. This implementation allows for fast $O(1)$ lookups at run-time, at the expense of higher memory usage.

Secondary features are determined in the context of a hostname and location candidate pair. As shown in Figure 7, we first determine all candidates before we can compute the secondary features. An example of secondary features for the *Rochester, MN* candidate is *Admin1 Match*, which is *true* only if the administrative region of the candidate location can be found in a different term of the hostname. Since the hostname contains the term `mn`, which is an abbreviation of Minnesota, then this secondary feature is *true* for the first candidate. However, it is *false* for the second candidate, because *Rocha* is in an administrative region also called *Rocha*, and it cannot be found in the hostname. *First Letters Admin 1* is similar, but it matches at least 3 first consecutive letters of administrative names. *Country* and *Country TLD* both try to match the country of the current candidate by searching for a country code in the hostname terms or in the *TLD*, respectively.

5.3 Classifier

For a given hostname, our reverse DNS geolocation can extract and evaluate tens of potential location candidates. For example, if one of the terms of the hostname is `york`, the initial list of candidates will contain all locations named *York* in the world. We run a binary classifier on each of the initial candidates. The classifier uses the primary and secondary features to evaluate if it is plausible for the hostname to be located in a candidate location. All the candidates where the classifier returns `false` are discarded. The remaining plausible candidates are sorted by confidence and returned in a list.

Although exhaustively determining the optimal type of binary classifier is outside of the scope of this work, we tested four variations of the classifier: logistic regression, C4.5 decision trees, random forest, and SVM. Logistic regression had the best performance on a small validation set. Consequently, we performed all experiments in Section 6 using this classifier.

During training the classifier learns to determine if any given hostname and candidate location pair is plausible. Since we are using a binary classifier and decision trees, the learned rules would look like: "if a part of the hostname matches the given candidate city, and there is another part of the hostname which also matches the administrative region where the city is located, then it is plausible that the hostname is located in the given candidate city (output is `true`)". Note that the classifier rules themselves do not contain locations. Another example rule is: "if the hostname contains a CLLI code that matches the candidate city, and if the hostname TLD matches the country of the city, then it is plausible that the hostname is located in the given city (output is `true`)". If for a given hostname we have two location candidates, for example if one that matches the first rule and one that matches the second, we choose the candidate where the rule has higher confidence. Each rule is given a confidence during training.

5.4 Sampling Strategy

We propose sampling the training set to account for data bias, to improve generalization, and to reduce the amount of required training data. First, the entire set of reverse DNS hostnames is naturally skewed towards the largest Internet Service Providers, which own the most addresses. Second, some feature categories such as *City Name* occur much more often than others such as *Abbreviations*. This can lead the classifier to ignore less frequent features categories. Third, during training multiple location candidates can be generated for each hostname, out of which at most one can be correct. Since the classifier is trained on hostname and candidate pairs, this also introduces another type of bias where the number of negative samples significantly outweighs the number of positive ones. Therefore, we sample data to account for some of this bias and to improve generalization through increased training data diversity.

We perform stratified sampling on the logical domains of the hostnames, keeping at most $\mathcal{X}$ samples per logical domain. This approach ensures that naming schemes of large organizations do not significantly skew the training data. We further increase feature diversity by keeping a ratio of $\mathcal{Y} : 1$ between the number of samples that contain the most commonly occurring feature and the ones that contain the least occurring feature. Finally, we also enforce a ratio of $\mathcal{Z} : 1$ between the number of negative and positive examples. We evaluate our data sampling strategy and its parameters in Section 6.2.

5.5 Feature Importance

The coefficients of the logistic regression model can be used as a rough estimate of feature importance. These values that the most important primary features are host pattern, CLLI code match, No Vowels Name, and Airport Code. The most important secondary features are Country TLD match and Admin 1 (administrative region) letters match. This analysis reveals a couple of insights into the model. First, both primary and secondary features are useful for classification. Second, ISPs often use names that are derived from but do not exactly match the original city names.

We also separately investigated the importance of converting International Domain Names (IDN) to Unicode, as we previously described in Section 5.1. We found that, at least for our task and dataset, disabling IDN translation does not make much of a difference (less than 1% difference in all dimensions). It may be because not many IDN domains are used to name reverse DNS hostnames, as opposed to forward DNS (website domains).

6 EVALUATION

We evaluate our approach against three state-of-the-art academic baselines that also use hostname location hints, and two commercial geolocation databases which combine multiple geolocation approaches. We show that our method significantly outperforms academic baselines and is complementary and competitive to commercial databases.

6.1 Ground Truth

Our ground truth dataset contains 67 million randomly sorted IP addresses with known IP location, of which we used 40 million for training and 27 million for testing. We compiled the dataset in March 2018 from Bing query logs. It was derived from devices with global positioning sensors, where users opted-in to provide location information. These users agreed to share their location automatically at query time in order to receive personalized local results. The dataset contains both mobile and fixed broadband IP addresses, since users often connect their mobile devices to their home Wi-Fi. The location distribution of these addresses roughly follows that of worldwide Internet penetration. We determined the location of each IP address in the dataset using an automated pipeline which aggregated all locations reported for an individual IP address. IP addresses with a large variance in reported locations were removed as outliers. That is, we discarded any IP address that was present in multiple cities over the course of a month. This filtering step ensures we retain only mobile IP addresses that are contained within a single city, as well as fixed (Wi-Fi) IP addresses. At no point did we have direct access to the raw locations of individual users.

Since we sampled these IP addresses randomly from Bing search logs, not all of these IPs have reverse DNS hostnames, and not all the hostnames contain location hints. We have shown in Table 2 that only 33.4% of IPv4 addresses (1.24 billion) have a valid reverse DNS hostname, and only a further fraction contain hints. However, note that to build our classifier we also need examples of hostnames without location hints.

6.2 Preliminary Evaluation

We conducted two experiments to evaluate the binary classifier in isolation. In the first experiment, we randomly selected 100,000 IP addresses from the training set and performed ten-fold cross validation. We did not further sample the data in any other way. For each hostname, we extracted location candidates, then ran the binary classifier on all the pairs between the target hostname and each of its candidates. Since **our approach can return multiple plausible locations for a given hostname**, we choose the candidate with the highest classifier confidence. We break ties by selecting the location with the highest population, as a proxy of popularity. Table 7 contains definitions of the terms we use for evaluation in this section.

We obtained an accuracy of 99% for the binary classes, mostly because the vast majority of results were true negatives. We define accuracy as the total number of correct predictions divided by the total number of predictions made for a dataset. However, the true positive rate was only 67.6%, precision was 80.9%, and recall was 67.6%.

In the second experiment we introduced training data sampling as described in Section 5.4. We set the $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ parameters to *200*, *10*, and *3*, respectively. We again performed ten-fold cross validation. Although accuracy decreased to 92.9%, we obtained better results for true positive rate, precision, and recall, at 78.8%, 88.5%, and 78.8%, respectively. We varied the values of the $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ parameters using brute-force search but this did not alter the results significantly. The simple

Table 7. Acronyms and definitions used for the evaluation results of the binary classifier

Name	Description
TP	**True Positive** - Given a hostname and a candidate city, the binary classifier returned true and correctly determined that the hostname is in that city.
FP	**False Positive** - Given a hostname and a candidate city, the binary classifier returned true and incorrectly determined that the hostname is in that city.
FN	**False negative** - Given a hostname and a candidate city, the binary classifier returned false but the hostname was actually located in that city.
TN	**True negative** - Given a hostname and a candidate city, the binary classifier returned false and correctly determined that the hostname is NOT in that city.
Precision	**TP** / (**TP** + **FP**) - The number of positive class predictions that actually belong to the positive class.
Recall	**TP** / (**TP** + **FN**) - The number of positive class predictions made out of all positive examples in the dataset
Accuracy	(**TP** + **TN**) / (**TP** + **FP** + **FN** + **TN**) - The total number of correct predictions divided by the total number of predictions made for a dataset.

fact of introducing a stratified sampling strategy is what made the results improve, not necessarily how we varied its parameters. In conclusion, our sampling strategy helps the classifier generalize and it significantly improves results.

6.3 Academic baselines

We next perform an evaluation against three state-of-the-art academic baselines. Similarly to our approach, they receive a hostname as input and attempt to extract its location. The *undns* baseline from University of Washington consists of manually generated rules that map hostname patterns to locations [56]. The *DRoP* baseline from CAIDA at University of California relies on automatically generated rules derived from hostname patterns and validated by active measurement data (traceroutes) [33]. Finally, the *DDec* baseline also from CAIDA combines the results from *undns* and *DRoP* [21].

Although the baselines are offered as HTTP APIs, it is unclear if their underlying rules are kept up to date. This may impact the results in our comparison. We further discuss this potential issue in Section 9. In contrast, since our approach is completely automated it can be refreshed without manual intervention. Furthermore, we have shown at the bottom of Table 2 that only 1% of IPs with reverse DNS hostname are Internet infrastructure (router) IPs. Therefore, the vast majority of addresses are end-user IPs. The DRoP component of DDec was trained and evaluated specifically on router hostnames, therefore it is expected that its coverage for end-user IPs is poor.

Since all three baselines are available from a public web endpoint [21], we had to restrict the number of requests we made to a manageable size, out of politeness. For testing we initially selected multiple service providers of different sizes, spanning various countries around the world. However, the baselines were missing **any** rules for several of these providers, including *airtelbroadband.in* from India, *bigpond.net.au* from Australia, and *megared.net.mx* in Mexico. Although the baselines have good rule coverage in North America, they are at least partially lacking in international coverage. In the interest of fairness, we ended up with a list of eight providers, each of which are covered by at least two of the baselines.

To train the classifier, our sampling strategy only considered approximately 60,000 data points out of the 40 million hostnames in our training set. From our test set of 27 million IP addresses, we selected all of the ground truth data points which intersected the eight target providers, which yielded a testing subset of 1.6 million hostnames. We issued these requests to the CAIDA web endpoint and parsed the responses from each of the baselines.

Table 8 lists the results for all eight domains, as well as the overall results across the entire testing subset. Our approach is labeled *RDNS* in the table. We define the **error distance** in kilometers to be the distance between where a model places the location of a hostname, and the actual location of the IP address behind that hostname. Since both our approach and the baseline have city-level granularity, we use the locations of city centers. The first block of results shows median error distance in kilometers. We observed that **our model significantly outperforms the baselines** and its results are generally more stable across all domains. We also observed that the median error distance for several domains is abnormally high for the *DRoP* baseline, and sometimes for the other baselines as well. To further investigate this surprising finding we manually verified a small sample of results. Table 9 lists examples of locations extracted incorrectly by the *DRoP* baseline. In the last column of the table we list the rule that caused the incorrect extraction. For example, *DRoP* incorrectly determines that the hostname `d49-194-53-51.meb1.`**`vic`**`.optusnet.com.au` is in Vicenza, IT, using the rule `%<<iata>>.optusnet.com.au`. Although the *IATA* airport code *vic* is indeed located in Vicenza, the correct location is Melbourne, Victoria. We could not find any `optusnet.com.au` hostname where the rule was correct. In conclusion, the *DRoP* baseline contains incorrect rules for many domains. The results for the *undns* baseline also indicate high error distance for multiple test domains. After investigating the results, we found that *undns* sometimes maps entire TLDs to a single city. For example, the locations for all `163.data.com.cn` hostnames are extracted as *Beijing, CN*. Lastly, since *DDec* is a combination of *undns* and *DRoP*, it is also affected by incorrect rules.

We manually investigated cases where our approach yielded incorrect locations and we found three main reasons that lead to failures. First, when matching location hints are short, for example *nwmd*, it is difficult even for a human to determine the correct location. Second, sometimes the hostnames contain a city name that is ambiguous, for example *portland*, and does not contain any other supporting evidence. Since there are several cities called Portland, our approach picks the one with the highest population, which may not be correct. Third, we did find a few isolated cases where an ISP reallocated an IP address to a different city but did not update the hostname to contain the new city name.

The advantage of using median as a metric is that it is impervious to outliers, which can favor our model that can place false positives very far from the actual location, generating more outliers. To fairly characterize the results, we also computed *RMSE*, a metric at the other extreme of the spectrum. *RMSE*, which stands for root mean square error, easily gets swayed by large outliers. This poses a disadvantage for our model. We compute it using the error distance in kilometers for each hostname. The *RMSE* results in Table 8 show that generally our approach still outperforms the baselines in 6 out of 8 domains. In the two cases where our model has higher *RMSE* than the models, the coverage of our model is higher.

In 3 out of 8 cases the *undns* baseline has 100% coverage. We define coverage as the total number of hostnames where a model made a decision, over the total number of hostnames in the test set. *undns* having high coverage is a side effect of it using catch-all rules that map entire TLDs to a single city. In all three cases this leads to poor results for both median error and *RMSE*.

We define the combined score as the inverse of *RMSE* multiplied by coverage. As error distance improves (gets smaller), the combined score increases, and vice versa. Similarly, higher coverage

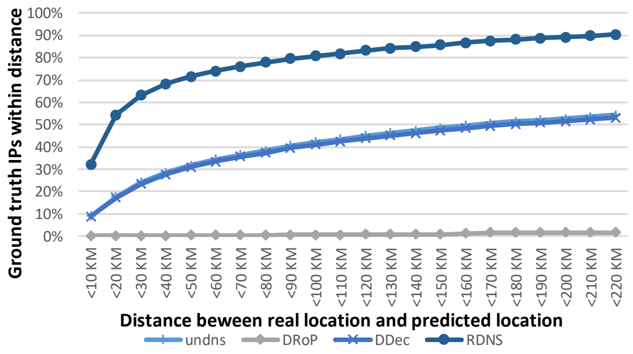


Fig. 8. Academic evaluation comparing our approach RDNS against three state-of-the-art academic baselines. Since the results for the undns and DDec baselines are similar, their curves are overlayed.

also improves the combined score, and vice versa. **Our approach significantly outperforms all academic baselines** when considering the combination of error distance and coverage.

Finally, Figure 8 displays the cumulative error distance in kilometers. The X axis represents the maximum distance between the real location and the predicted location. The Y axis shows how many hostnames and their IP addresses fall within the error distance. For instance, the *<20 km* column shows that our method labeled *RDNS* places approximately 54% of hostnames in the ground truth set within 20 kilometers of their actual location. Our method outperforms the baselines by a large margin. The *DRoP* baseline yields the worst results, underperforming the other methods.

6.4 Commercial baselines

In this work we focus on improving reverse DNS geolocation, which is only one source of geolocation information. Table 2 reveals that about a third of IP addresses have reverse DNS hostnames. A further subset of these hostnames contain location hints. While this can result in hundreds of millions of hostnames with location information, this is insufficient to cover the *IPv4* space.

Commercial geolocation databases combine and conflate multiple geolocation data sources. Information from reverse DNS hostnames is helpful but not sufficient to compile a full geolocation database. Our approach, which can output multiple location candidates for a given hostname, lends itself to being combined with other data sources to form a more complete database.

Although reverse DNS geolocation on its own cannot match the coverage of commercial databases which combine multiple techniques based on network delay, WHOIS database parsing, reverse DNS parsing, and paid data contracts, we compared our single technique against two commercial baselines (*Provider A* and *Provider B*) to show that it can complement and potentially improve these commercial offerings. **Provider A is MaxMind GeoLite2**. Provider B is another leading commercial geolocation service, but due to contractual obligations that disallow comparative benchmarking we cannot disclose its name. Our approach is entitled *RDNS* in the graphs. The first four graphs in Figure 9 show that for certain ISPs our approach outperforms, and thus is complementary and can be used to improve, commercial databases. To obtain these results we used the subset of the testing data which matches the domain of that particular ISP. For Bigpond, OCN, Megared, and Qwest the subsets correspond to 69,993, 92,665, 94,972, and 398,194 testing IP addresses, respectively.

Table 8. Evaluation of our approach *RDNS* against three state-of-the-art academic baselines: *undns*, *DRoP*, and *DDec*.

Metric →		**Median Error in km** (lower is better)				**RMSE based on km** (lower is better)				**Coverage** (higher is better)				**Combined score** (higher is better)			
Domain ↓	#	**undns**	**DRoP**	**DDec**	**RDNS**	**undns**	**DRoP**	**DDec**	**RDNS**	**undns**	**DRoP**	**DDec**	**RDNS**	**undns**	**DRoP**	**DDec**	**RDNS**
163data.com.cn	166K	1,517.5	N/A	1,517.5	**10.6**	1,495	N/A	1,495	**404**	**100%**	N/A	**100%**	94.5%	0.67	N/A	0.67	**2.34**
bell.ca	200K	N/A	5,875.2	5,875.2	**6.0**	N/A	5,807	5,807	**1,262**	N/A	2.3%	2.3%	**95.7%**	N/A	0.00	0.00	**0.76**
brasiltelecom.net.br	32K	808.7	5,628.7	808.7	**15.2**	889	5,620	889	**427**	**100%**	69.7%	**100%**	73.9%	1.12	0.12	1.12	**1.73**
charter.com	580K	60.8	N/A	60.8	**59.9**	**478**	N/A	**478**	484	78.0%	N/A	78.0%	**89.0%**	1.63	N/A	1.63	**1.84**
frontiernet.net	67K	36.5	6,247.6	36.5	**16.7**	785	6,101	785	**689**	3.6%	0.8%	3.6%	**99.4%**	0.05	0.00	0.05	**1.44**
nttpc.ne.jp	0.9K	9.5	9,259.9	16.2	**9.1**	**2,081**	9,161	4,976	3,694	12.0%	16.2%	16.2%	**57.6%**	0.06	0.02	0.03	**0.16**
optusnet.com.au	100K	704.4	16,134.6	704.4	**12.7**	1,175	16,374	1,175	**583**	**100%**	49.8%	**100%**	98.9%	0.85	0.03	0.85	**1.70**
qwest.net	408K	3,426.6	8,038.7	8,038.7	**17.6**	6,856	7,361	7,361	**427**	0.0%	4.1%	4.1%	**94.0%**	0.00	0.01	0.01	**2.20**
Overall	1.6M	163.9	13,974.2	177.9	**17.5**	924.0	12,640.4	1,497.5	**677.8**	48.3%	6.1%	49.7%	**92.3%**	0.52	0.00	0.33	**1.36**

Table 9. Examples of locations extracted incorrectly by the *DRoP* baseline, compared to the correct locations extracted by our approach

Hostname	**Incorrectly Extracted Location (baseline)**	**Correct Location (our approach)**	**Incorrect DRoP Baseline Rule**
malton2259w-lp140-03-50-100-186-228.dsl.bell.ca	**malton** → Malton, North Yorkshire, England	**malton**, **.ca** → Malton, Canada	%<<pop>>([^L]+L+D*){3}.bell.ca
200-96-182-198.cbace700.**dsl**.brasiltelecom.net.br	**dsl** → Daru, Sierra Leone	**cbace**, **.br** → Cuiabá, Brazil	%<<iata>>.brasiltelecom.net.br
70-100-143-28.dsl2-**pixley**.roch.ny.frontiernet.net	**pixley** → Pixley, California, USA	**roch**, **ny** → Rochester, New York, USA	%<<pop>>([^L]+L+D*){2}.frontiernet.net
st0120.**nas**931.m-hiroshima.nttpc.ne.jp	**nas** → Nassau, Bahamas	**hiroshima**, **.jp** → Hiroshima, Japan	%<<iata>>([^L]+L+D*){2}.nttpc.ne.jp
d49-194-53-51.meb1.**vic**.optusnet.com.au	**vic** → Vicenza, Italy	**meb**, **vic**, **.au** → Melbourne, Victoria	%<<iata>>.optusnet.com.au
71-209-14-48.**bois**.qwest.net	**bois** → 's-Hertogenbosch, The Netherlands	**bois** → Boise, Idaho, USA	%<<pop>>.qwest.net

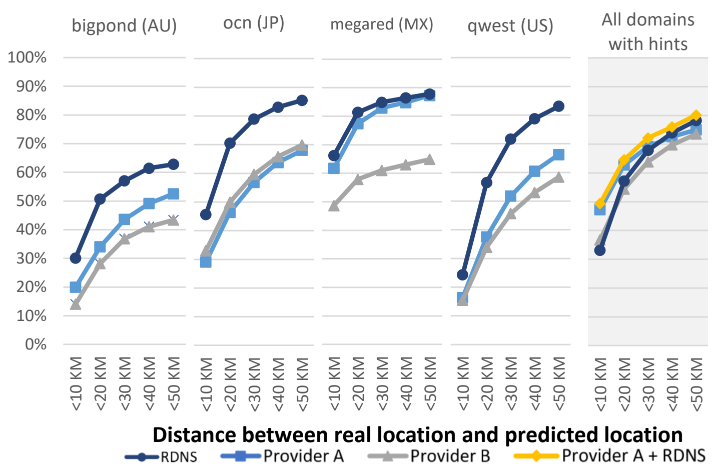


Fig. 9. Commercial evaluation which compares our approach RDNS against two state-of-the-art commercial IP geolocation providers. The first four graphs in the figure show results for specific domains, while the last graph shows overall results.

To generate the fifth graph in the figure we ran our system using a two-step approach. In the first step we iterated over all hostnames and automatically determined the domains which often contained location hints, by computing the distribution of features and picking the domains where the majority of hostnames contained at least one primary feature. In the second step, we evaluated our technique on those domains, keeping the same separation of training and testing data at each domain level. This yielded a dataset of 1.9 million IP addresses. The results show that our approach is competitive with commercial geolocation services, when the domains contain location hints. Comparatively our approach yields good results even though we use a single IP geolocation method while the baselines combine multiple techniques. The median error for our approach is 16.5 kilometers, while for the commercial providers *A* and *B* it is 11.1 and 16.7 kilometers, respectively.

To further demonstrate that our approach is complementary with commercial databases, in the fifth graph of Figure 9 we also display results for a preliminary hybrid approach called *Provider A + RDNS*. To compile this combined database we replaced the locations predicted by *Provider A* with the locations determined using our method, whenever our classifier has a confidence above 85%. Results show that the hybrid approach outperforms both our original approach, and the two commercial providers, with a median error of 10.3 kilometers.

7 PRIVACY

Online privacy is becoming increasingly important. In 2016, Pew Research found that while many Americans are willing to share personal information to access online services, they are often cautious about disclosing their information and are frequently unhappy about what happens to that information once companies have collected it [48]. In a separate 2013 study, Pew Research has also found that almost half of teen app users have turned off location tracking on their phone, because they are worried about other people or companies being able to access that information [9]. Over a third of adult smartphone users have done the same.

In 2018, a *New York Times* exposé revealed the widespread industry practice of tracking the precise location of people through smartphone apps [59, 62]. They reported that at least 75 companies such as Reveal Mobile [41], SafeGraph [52], Kiip [36], and Fysical [25] collect location data through local news and weather forecast apps. More than 1,000 popular apps contain location-sharing code from such companies. Several of these companies claim to track up to 200 million mobile devices in the United States, which is half of all the phones used in 2017. While seemingly anonymous, location information distributed by these companies is in fact hyper-local and extremely granular, which allowed reporters to easily uncover the identities of multiple people based only on their movements.

Security researchers at AppCensus have similarly found in 2018 close to 2,000 apps that send out location data, or Wi-Fi router MAC addresses that can be converted into location data with the help of other databases. More than 150 of these apps were targeted at children [51].

We have designed both our approach and our evaluation with this sensitive subject in mind. Our proposed geolocation method relies on reverse DNS hostnames shared publicly by Internet Service Providers. These hostnames provide only coarse city-level or region level location information. Therefore our approach may be more privacy conscious than the widespread industry practice of requesting exact GPS coordinates through mobile apps or the HTML5 Geolocation API. Furthermore, we have anonymized our ground truth set by modifying raw locations in a random direction by 584 meters, aggregating all locations reported for an IP address, and reducing location accuracy to city-level. We did not have access to the raw locations of individual users at any point in this automated process.

8 REPRODUCING RESULTS

To aid in reproducing and extending our results, we are making available the major components of our approach:

- **Hostname splitter**: This module splits hostnames into their constituent parts. It also includes the terms blacklist.
- **Sampling strategy**: This component performs stratified sampling during training.
- **Features**: We are also open sourcing the primary and secondary features generators. For feature generation we have purposely used publicly available datasets as described in Section 3.
- **Classifier**: The actual classifier, including the training and testing pipeline. In addition to the source code, we are also including a binary trained version of the classifier.

The code is available at **github.com/microsoft/ReverseDNSGeolocation**.

9 LIMITATIONS

Although our proposal is promising and could be an important part of an IP geolocation service, it also has limitations:

- **IP coverage**: In Table 2 we have shown that only 33.4% of IPv4 addresses (1.24 billion) have a valid reverse DNS hostname. This intrinsic limitation means that our approach alone cannot be used to compile a full geolocation database. Instead, one would have to combine it with other complementary data sources in order to match or surpass the coverage of commercial geolocation databases, which use multiple techniques. Furthermore, even when an IP has a reverse DNS hostname, it does not mean the hostname contains location hints, which is a further inherent limitation. Using a preliminary naïve matching technique we have demonstrated that only 14.1% of these hostnames could contain exact city matches, and only 23.5% of them could contain airport code matches.
- **Name ambiguity**: Location names in hostnames are often shortened or abbreviated. Although our approach does a reasonable job of matching vague name and disambiguating among multiple potential location candidates, it still sometimes does not have enough supporting evidence to

choose among multiple plausible location candidates. With this limitation in mind, we have designed our approach to be combined with other data sources that would help further disambiguate among the candidates.

- **Incorrect hints**: Even if we are able to extract a clear location from a hostname, it still does not mean that the IP address is actually in that location. Reverse DNS hostname entries are managed by the ISP themselves, or sometimes their clients, and can become outdated. Although we were not able to find widespread problems with stale records across entire ISPs, we were able to find individual hostnames where the ground truth location did not match the location in the hostname.
- **Focus on end-user IPs**: As shown at the bottom of Table 2, our ground truth data covers mainly end-user IP addresses, while one of the academic baselines (DRoP [33]) was originally only evaluated on infrastructure IPs. However, typically ISPs use the same hostname naming scheme for both router addresses and end-user addresses. Furthermore, our commercial baselines are typically evaluated against end user-IP addresses. Last but not least, most commercial geolocation databases are used in practice against end-user IP addresses, as described in Section 1.
- **Academic baseline staleness**: We use three academic baselines: undns [56], DRoP [33], and *DDec* [21], with the latter being a combination of the first two. Work on undns was published in 2012, while a description of DRoP was published in 2014. Finally, the first paper we found which describes DDec is from 2016 [13]. When interpreting the evaluation results please keep in mind that our approach is fresher. It is possible the baselines would fare better against our proposal if they would be more up to date.

10 CONCLUSIONS AND FUTURE WORK

We presented a machine learning approach to IP geolocation using reverse DNS hostnames. Our method significantly outperforms several state-of-the-art academic baselines, with a median error of only 17.5 kilometers for our approach when compared to 163.9, 13,974.2, and 177.9 kilometers (lower is better) for the three baselines respectively, a RMSE of 677.8 for our technique compared to 924.0, 12,640.4, and 1,497.5 (lower is better) for the three baselines respectively, and a coverage of 92.3% when compared to 48.3%, 6.1%, and 49.7% (higher is better) for the baselines, respectively.

Our proposal successfully tackles the challenges described in Section 1 by introducing two levels of features. Our primary features are designed to match acronyms, name variations, and different industry codes for locations such as airports, CLLI, and UN/LOCODE. We also defined a novel feature called host patterns which, given enough training data, captures the relationships of strings to locations that were not covered by the other primary features. We use secondary features such as country TLDs and administrative codes to buttress and further disambiguate between multiple location candidates. If at the end there are multiple candidates with the same confidence, we pick the one with highest population.

Compared to previous approaches such as *GeoTrack* [46] and *Undns* [56], our proposal does not require any human annotation or intervention. This means that given enough training data, it can automatically learn rules that were previously handcrafted by humans. Therefore, our approach makes it easier to scale to worldwide ISPs. Furthermore, our approach can also evaluate hostnames with domains that it has never seen before. Whereas *Undns* and *GeoTrack* had fixed rules per ISP domain, most features in our classifier are not domain specific, and therefore the classifier is resilient to variations of hostnames within the same domain and across domains. Other academic systems such as *DRoP* [33] and *DDec* [21](which is derived from DRoP) do use rules learned automatically. However, our evaluation has shown that the rules learned by *DRoP* are often not correct, as can be seen in Table 9, which leads to poor results.

Furthermore, our approach is complementary to and competitive with commercial baselines. We showed that there are several domains where our approach outperforms two state-of-the-art commercial databases, which demonstrates that our proposal is complementary and could be combined with these commercial baselines to improve them. We also show that on domains that contain location hints we are competitive with the commercial services, our approach having a median error of 16.5 kilometers, when compared to 11.1 and 16.7 kilometers for the commercial baselines, respectively. This is a promising result, when considering that here we focus on a single type of IP geolocation technique, whereas commercial databases combine multiple approaches.

Finally, one advantage of our approach is that it can output multiple plausible locations for a single hostname, in case of ambiguity. It thus lends itself to being combined with other data sources to form a more complete geolocation database. Future work could focus on combining reverse DNS hostname information with WHOIS databases and network delay to form a more complete geolocation database that could be more competitive with commercial offerings.

REFERENCES

[1] 2017. *IPv4 Special-Purpose Address Registry.* Technical Report. Internet Assigned Numbers Authority. https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml Accessed: 2018-06-27.

[2] John Akhilomen. 2013. Data Mining Application for Cyber Credit-Card Fraud Detection System. In *Advances in Data Mining. Applications and Theoretical Aspects*, Petra Perner (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 218–228.

[3] Lars Backstrom, Eric Sun, and Cameron Marlow. 2010. Find Me if You Can: Improving Geographical Prediction with Social and Spatial Proximity. In *WWW 2010.* ACM, Raleigh, North Carolina, USA, 61–70. https://doi.org/10.1145/1772690.1772698

[4] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. 2011. Inferring and Using Location Metadata to Personalize Web Search. In *SIGIR 2011.* ACM, Beijing, China, 135–144. https://doi.org/10.1145/2009916.2009938

[5] Tej Paul Bhatla, Vikram Prabhu, and Amit Dua. 2003. Understanding credit card frauds. *Cards business review* 1, 6 (2003).

[6] R. Braden. 1989. *Requirements for Internet Hosts – Application and Support.* RFC 1123. RFC Editor. https://tools.ietf.org/html/rfc1123

[7] Asmir Butkovic, Fahrudin Orucevic, and Anel Tanovic. 2013. Using whois based geolocation and google maps api for support cybercrime investigations. In *WSEAS International Conference on Circuits, Systems, Communications, Computers and Applications (CSCCA'13).* 194–201.

[8] CAIDA. 2018. The CAIDA Internet Topology Data Kit - 2018-03. https://www.caida.org/data/internet-topology-data-kit Accessed: 2020-08-19.

[9] Pew Research Center. 2013. Location-Based Services. http://www.pewinternet.org/2013/09/12/location-based-services/ Accessed: 2019-02-06.

[10] Joseph Chabarek and Paul Barford. 2013. What's in a Name? Decoding Router Interface Names. In *Proceedings of the 5th ACM Workshop on HotPlanet (HotPlanet '13).* Association for Computing Machinery, New York, NY, USA, 3–8. https://doi.org/10.1145/2491159.2491163

[11] Balakrishnan Chandrasekaran, Mingru Bai, Michael Schoenfield, Arthur Berger, Nicole Caruso, George Economou, Stephen Gilliss, Bruce Maggs, Kyle Moses, David Duff, et al. 2015. Alidade: Ip geolocation without active probing. *Department of Computer Science, Duke University, Technical Report* (2015).

[12] Gloria Ciavarrini, Maria S Greco, and Alessio Vecchio. 2018. Geolocation of Internet hosts: Accuracy limits through Cramér–Rao lower bound. *Computer Networks* 135 (2018), 70–80.

[13] Kc Claffy. 2016. The 7th Workshop on Active Internet Measurements (AIMS7) Report. *ACM SIGCOMM Computer Communication Review* 46, 1 (2016), 50–57.

[14] Ovidiu Dan, Vaibhav Parikh, and Brian D. Davison. 2016. Improving IP Geolocation Using Query Logs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM '16).* Association for Computing Machinery, New York, NY, USA, 347–356. https://doi.org/10.1145/2835776.2835820

[15] Ovidiu Dan, Vaibhav Parikh, and Brian D Davison. 2018. Distributed Reverse DNS Geolocation. In *2018 IEEE International Conference on Big Data (Big Data).* IEEE, 1581–1586.

[16] Digital Element. 2018. Finding Yourself: The Challenges of Accurate IP Geolocation. https://dyn.com/blog/finding-yourself-the-challenges-of-accurate-ip-geolocation/ Accessed: 2019-02-06.

[17] Ben Du, Massimo Candela, Bradley Huffaker, Alex C Snoeren, and kc claffy. 2020. RIPE IPmap active geolocation: mechanism and performance evaluation. *ACM SIGCOMM Computer Communication Review* 50, 2 (2020), 3–10.

[18] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. 2013. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *22nd USENIX Security Symposium (USENIX Security 13)*. USENIX Association, Washington, D.C., 605–620. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric

[19] H. Eidnes, G. de Groot, and P. Vixie. 1998. *Classless IN-ADDR.ARPA delegation*. RFC 2317. RFC Editor. https://tools.ietf.org/html/rfc2317

[20] P.T. Endo and D. Sadok. 2010. Whois Based Geolocation: A Strategy to Geolocate Internet Hosts. In *AINA 2010*. 408–413. https://doi.org/10.1109/AINA.2010.39

[21] Center for Applied Internet Data Analysis. [n. d.]. DDec - DNS Decoded - CAIDA's public DNS Decoding database. http://ddec.caida.org/help.pl Accessed: 2018-07-31.

[22] United Nations Economic Commission for Europe. [n. d.]. UN/LOCODE: United Nations Code for Trade and Transport Locations. https://www.unece.org/cefact/locode/welcome.html Accessed: 2018-06-27.

[23] Mozilla Foundation. [n. d.]. Public Suffix List. https://publicsuffix.org/list/ Accessed: 2018-06-28.

[24] Michael J. Freedman, Mythili Vutukuru, Nick Feamster, and Hari Balakrishnan. 2005. Geographic locality of IP prefixes. 153–158. 5th ACM SIGCOMM Conference on Internet Measurement, IMC 2005 ; Conference date: 19-10-2005 Through 21-10-2005.

[25] Fysical. 2019. The next data frontier isn't digital. It's Fysical. https://fysical.org/ Accessed: 2019-02-06.

[26] Manaf Gharaibeh, Anant Shah, Bradley Huffaker, Han Zhang, Roya Ensafi, and Christos Papadopoulos. 2017. A Look at Router Geolocation in Public and Commercial Databases. In *Proceedings of the 2017 Internet Measurement Conference (IMC '17)*. Association for Computing Machinery, New York, NY, USA, 463–469. https://doi.org/10.1145/3131365.3131380

[27] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. 2006. Constraint-Based Geolocation of Internet Hosts. *IEEE/ACM Transactions on Networking* 14, 6 (Dec 2006), 1219–1232. https://doi.org/10.1109/TNET.2006.886332

[28] Chuanxiong Guo, Yunxin Liu, Wenchao Shen, H.J. Wang, Qing Yu, and Yongguang Zhang. 2009. Mining the Web and the Internet for Accurate IP Address Geolocations. In *INFOCOM 2009*. 2841–2845. https://doi.org/10.1109/INFCOM.2009.5062243

[29] Aniko Hannak, Piotr Sapiezynski, Arash Molavi Kakhki, Balachander Krishnamurthy, David Lazer, Alan Mislove, and Christo Wilson. 2013. Measuring Personalization of Web Search. In *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*. Association for Computing Machinery, New York, NY, USA, 527–538. https://doi.org/10.1145/2488388.2488435

[30] K Harrenstien, M Stahl, and E Feinler. 1985. *DoD Internet host table specification*. RFC 952. RFC Editor. https://tools.ietf.org/html/rfc952

[31] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. 2000. Algorithms for association rule mining—a general survey and comparison. *ACM sigkdd explorations newsletter* 2, 1 (2000), 58–64.

[32] Cheng Huang, D.A. Maltz, Jin Li, and Albert Greenberg. 2011. Public DNS system and Global Traffic Management. In *INFOCOM 2011*. 2615–2623. https://doi.org/10.1109/INFCOM.2011.5935088

[33] Bradley Huffaker, Marina Fomenkov, and kc claffy. 2014. DRoP:DNS-based Router Positioning. *ACM SIGCOMM Computer Communication Review (CCR)* 44, 3 (Jul 2014), 6–13.

[34] Stephen Mark Huffman and Michael Henry Reifer. 2005. Method for geolocating logical network addresses. US Patent 6,947,978.

[35] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. 2006. Towards IP Geolocation Using Delay and Topology Measurements. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC '06)*. Association for Computing Machinery, New York, NY, USA, 71–84. https://doi.org/10.1145/1177080.1177090

[36] Kiip. 2019. Moments Based In-App Mobile Advertising. http://www.kiip.me/ Accessed: 2019-02-06.

[37] Chloe Kliman-Silver, Aniko Hannak, David Lazer, Christo Wilson, and Alan Mislove. 2015. Location, Location, Location: The Impact of Geolocation on Web Search Personalization. In *Proceedings of the 2015 Internet Measurement Conference (IMC '15)*. Association for Computing Machinery, New York, NY, USA, 121–127. https://doi.org/10.1145/2815675.2815714

[38] Bernhard Kölmel and Spiros Alexakis. 2002. Location Based Advertising. In *First International Conference on Mobile Business*. Athens, Greece.

[39] Lori MacVittie. 2012 (accessed August 2, 2018). Geolocation and Application Delivery. https://www.f5.com/pdf/white-papers/geolocation-wp.pdf.

[40] Douglas Maughan et al. 2009. A roadmap for cybersecurity research. *US Department of Homeland Security* 2009 (November 2009).

[41] Reveal Mobile. 2019. Win More Business with Location-Based Marketing & Analytics. https://revealmobile.com/ Accessed: 2019-02-06.

[42] P. Mockapetris. 1987. *DOMAIN NAMES - CONCEPTS AND FACILITIES*. RFC 1034. RFC Editor. https://tools.ietf.org/html/rfc1034

[43] James A Muir and Paul C Van Oorschot. 2009. Internet geolocation: Evasion and counterevasion. *Acm computing surveys (csur)* 42, 1 (2009), 4.

[44] Abdullah Yasin Nur and Mehmet Engin Tozal. 2018. Geography and Routing in the Internet. *ACM Trans. Spatial Algorithms Syst.* 4, 4, Article 11 (Sept. 2018), 16 pages. https://doi.org/10.1145/3239162

[45] A. Costello P. Faltstrom, P. Hoffman. 2003. *Internationalizing Domain Names in Applications (IDNA)*. RFC 3490. RFC Editor. https://tools.ietf.org/html/rfc3490

[46] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. 2001. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *SIGCOMM 2001*. ACM, San Diego, California, USA, 173–185. https://doi.org/10.1145/383059.383073

[47] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. 2011. IP geolocation databases: Unreliable? *ACM SIGCOMM Computer Communication Review* 41, 2 (2011), 53–56.

[48] Lee Rainie and Maeve Duggan. 2016. Privacy and information sharing. *Pew Research Center* (2016).

[49] Rapid7Labs. [n. d.]. Reverse DNS (RDNS) - 2013-2017. https://opendata.rapid7.com/sonar.rdns/. Accessed: 2018-06-23.

[50] Rapid7Labs. [n. d.]. *Reverse DNS (RDNS) v2 - 2017 onward*. Technical Report. https://opendata.rapid7.com/sonar.rdns_v2/ Accessed: 2018-06-23.

[51] Joel Reardon. 2018. Apps sending location, secretly. (2018). https://blog.appcensus.mobi/2018/05/14/apps-sending-location-secretly/ Accessed: 2019-02-06.

[52] SafeGraph. 2019. The Source of Truth for Physical Places. https://www.safegraph.com/ Accessed: 2019-02-06.

[53] Quirin Scheitle, Oliver Gasser, Patrick Sattler, and Georg Carle. 2017. HLOC: Hints-Based Geolocation Leveraging Multiple Measurement Frameworks. *arXiv preprint arXiv:1706.09331* (2017).

[54] Yuval Shavitt and Noa Zilberman. 2011. A geolocation databases study. *IEEE Journal on Selected Areas in Communications* 29, 10 (2011), 2044–2056.

[55] Craig A. Shue, Nathanael Paul, and Curtis R. Taylor. 2013. From an IP Address to a Street Address: Using Wireless Signals to Locate a Target. In *7th USENIX Workshop on Offensive Technologies (WOOT 13)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/woot13/workshop-program/presentation/shue

[56] Neil Spring, Ratul Mahajan, and David Wetherall. 2002. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review* 32, 4 (2002), 133–145.

[57] Dan Jerker B Svantesson. 2007. E-Commerce Tax: How The Taxman Brought Geography To The 'Borderless' Internet. *Revenue Law Journal* 17, 1 (2007), 11.

[58] Geo Targetly. 2019. Automatically Switching Website Language Based On Visitor Country. https://geotargetly.com/automatically-switch-website-language-based-on-country Accessed: 2019-01-26.

[59] The New York Times. 2018. How The Times Analyzed Location Tracking Companies. *New York Times, Dec* 10 (2018). https://www.nytimes.com/2018/12/10/technology/location-tracking-apps-privacy.html Accessed: 2019-02-06.

[60] Paul Timmins. [n. d.]. TelcoData Telecommunications Database. https://www.telcodata.us/ Accessed: 2018-06-27.

[61] Marketa Trimble. 2011. The future of cybertravel: legal implications of the evasion of geolocation. *Fordham Intell. Prop. Media & Ent. LJ* 22 (2011), 567.

[62] Jennifer Valentino-DeVries, Natasha Singer, Michael H Keller, and A Krolik. 2018. Your Apps Know Where You Were Last Night, and They're Not Keeping It Secret. *New York Times, Dec* 10 (2018). https://www.nytimes.com/interactive/2018/12/10/business/location-data-privacy-apps.html Accessed: 2019-02-06.

[63] Yong Wang, Daniel Burgener, Marcel Flores, Aleksandar Kuzmanovic, and Cheng Huang. 2011. Towards Street-level Client-independent IP Geolocation. In *NSDI 2011*. USENIX, Berkeley, CA, USA, 365–379. http://dl.acm.org/citation.cfm?id=1972457.1972494

[64] Lin Wei, Guoming Ren, Lei Shi, Yongcai Tao, and Yangjie Cao. 2013. How does the recursive undns algorithm affect the accuracy of an IP geolocation system?. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2013 10th International Conference on*. IEEE, 1060–1064.

[65] Marc Wick. [n. d.]. GeoNames. http://download.geonames.org/export/dump/ Accessed: 2018-06-27.

[66] Bernard Wong, Ivan Stoyanov, and Emin Gün Sirer. 2007. Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In *Proceedings of the 4th USENIX Conference on Networked Systems Design amp; Implementation (NSDI'07)*. USENIX Association, USA, 23.

[67] Inja Youn, Brian L. Mark, and Dana Richards. 2009. Statistical Geolocation of Internet Hosts. In *ICCCN 2009*. 1–6. https://doi.org/10.1109/ICCCN.2009.5235373